LUDWIG-MAXIMILIAN-UNIVERSITY OF MUNICH
Department "Institute for Informatics"
Teaching and Research Units Media Informatics
Prof. Dr. Heinrich Hußmann

Master Thesis

Multi-User Gaze Pursuits: Interaction & Visual Feedback

Huyen Linh Nguyen Vo
huyenlinh.nguyenvo@gmail.com

Working period:	December 1st, 2016 until June 1st, 2017
Supervisor:	Mohamed Khamis
Responsible Professor:	Prof. Dr. Florian Alt

Abstract

Gaze interaction without the need to calibrate is a current trend in the research on HCI. One of the resulting interaction methods is called *Smooth Pursuits*, which is based on the underlying concept. In addition to that the use of public displays for collaborative tasks, or multi-user interaction in general, is also a trend expanding throughout HCI research. If paired together, smooth pursuits for multiple users on collaborative displays require the respective group of users to obtain feedback by the computer interface, as this marks an essential step in showing multiple users their respective partner's interaction intent, as well as which part of the screen they currently are focused on. In this thesis we present a direct comparison of methods to provide visual feedback with not enabling any type feedback, on users' selection speed, the numbers of selection errors they make and the count of how many times they find themselves distracted by the given feedback, as well as to assess their qualitative opinions on the several feedback types. For that matter we prepare a repeated measures study, combined with a qualitative and quantitative assessment involving showing three types of visual feedback, no, normal and gradual, to multiple simultaneous users based on object pursuit selections. We further speculate ways to overcome given challenges that arise with the use of public displays, smooth pursuits and multi-user interactions.

Gaze Interaction ohne das Bedürfnis der Kalibrierung wird als aktueller Trend im Forschungsbereich der Mensch-Maschine-Interaktion betrachtet. Eine der daraus folgenden Interaktionstechniken wird *Smooth Pursuits* genannt, eine Bezeichnung, die auf dem dahinterliegenden Prinzip der Selektion beruht. Zudem wird der Einsatz von *Public Displays* für Interaktionen welche mehrere Nutzer, auch im Rahmen von kollaborativen Nutzungsmustern, umschließen, rege diskutiert. Wenn man beide Aspekte der MMI kombiniert, wird das Bedürfnis den einzelnen Nutzern während Interaktionsprozessen welche auf *Smooth Pursuits* in Anbetracht von mehreren gleichzeitig agierenden Nutzern auf demselben Bildschirm Feedback zu geben, immer wichtiger. Dies hilft den Gruppen dabei eine Ahnung zu erhalten, mit welchem Intent ihr jeweiliger Partner interagiert und in welchem Bereich des Bildschirms der aktuelle Fokus liegt. Teil dieser Masterarbeit ist es einen direkten Vergleich zwischen verschiedenen Feedback-Methoden herzustellen, auch im Bezug auf den Fall, wenn Nutzer kein visuelles Feedback während ihrer Interaktion erhalten. Dieser Vergleich wird in Hinbetracht auf Auswahlgeschwindigkeit, Fehlerrate und Ablenkungsfällen bei den Selektionen getestet. Außerdem wird die persönliche Meinung der Nutzer zu den einzelnen Feedback-Möglichkeiten erfragt. Um dies zu verwirklichen, wird im Folgenden eine Studie basierend auf wiederholten Messungen, in Kombination mit qualitativer und quantitativer Beurteilung von drei gezeigten Möglichkeiten visuelles Feedback bei Selektionsaufgaben zu geben, vorbereitet. Weiterhin werden Möglichkeiten spekuliert wie auftretende Herausforderungen bezüglich der Interaktion mehrerer Nutzer auf *Public Displays* in Kombination mit *Smooth Pursuits*, überwunden werden können.

Definition of Tasks

Task Description Public displays commonly expect multiple users interacting simultaneously. Gaze-based interaction, while being promising for public displays, has often been deployed in single user scenarios, where a single user interacts at a time. Advances in computer vision algorithms and falling hardware prices promise the availability of gaze-based interaction with public displays for multiple users in the near future.

Additionally, gaze interaction with public displays requires special techniques that can accommodate with the need for immediately usable systems. One of these techniques, which is also state of the art in gaze interaction with public displays, is Pursuits. To date, Pursuits has never been used for multiple users. While it is straight forward to implement Pursuits for multiple users, it is not clear how feedback can be provided in the case of multiple users interacting simultaneously via Pursuits.

To close this gap, the goal of this thesis is to explore alternatives for visual feedback when interacting via Pursuits on public displays.

Tasks

- Survey of related work.
- Implementing Pursuits for multiple users. This can be realized using computer vision algorithms that require RGB cameras, or by the use of multiple eye trackers.
- Investigating the design space of visual feedback via Pursuits
- Implementing a prototype that delivers different types of visual feedback, and evaluating it in a user study.
- Analyzing the results and discussing them

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Munich, May 31, 2017

. . . .

Contents

1 Introduction **1**

2 Related Work **3**
2.1 Public and collaborative displays 3
2.2 Challenges using Public Displays 4
2.3 Gaze Interaction 5
2.3.1 Gaze as means of Selection 6
2.3.2 Calibration vs. No Calibration 7
2.3.3 Pursuit Interaction 7
2.3.4 Feedback 8
2.4 Multiple User Interaction Using Gaze on Larger Displays 9

3 Concept Development **11**
3.1 First Decision: Gaze Tracking Method 12
3.2 Second Decision: Incorporating Gaze as Interaction Method 12
3.3 Third Decision: Type of Feedback to give to Users 12
3.4 Research Problem, Hypotheses and Methods 13
3.5 Real-Life Scenario based on the Concept 14

4 Tracking Gaze with an RGB Camera **17**
4.1 First Attempt: Obtaining Gaze 18
4.2 Second Attempt: Correlation 19
4.3 Third Attempt: Mid-Pupil-Point 20
4.4 Final Attempt: Post-Process Pursuit 21
4.5 Summary 22

5 Implementation **23**
5.1 List of Frameworks and Libraries 23
5.1.1 tobiisdk4j - A Wrapper to use the Tobii SDK with Java 23
5.1.2 Processing 3.0 - A Visual Arts Context Programming Language 23
5.1.3 Commons Math - The Apache Commons Mathematics Library 24
5.1.4 Minim - An Audio Library for Processing 24
5.2 Implementation Process 24
5.2.1 First Milestone: Tracking Gaze for Multiple Users 25
5.2.2 Second Milestone: Moving Circles as Stimuli 26
5.2.3 Third Milestone: Correlating Object and Gaze Movement 28
5.2.4 Fourth Milestone: Adding Feedback 30
5.2.5 Fifth Milestone: Prepare Logging Process 33
5.2.6 Sixth Milestone: Loading Study Tasks 34

6 User Study **37**
6.1 Study Design 37
6.2 Technical Setup 39
6.3 Study Procedure 40

7 Evaluation And Results **43**
7.1 Speed: Mean and Standard Deviation 44
7.1.1 First Attempt: Initial Mean and Standard Deviation 44
7.1.2 Second Attempt: Eliminating Outlier 45
7.2 Error Rate: Mean and Standard Deviation 47

7.3 Distraction Count: Mean and Standard Deviation 48
7.4 Comparing User Estimations with System Logs 49
7.5 ANOVA Significance Tests 51
7.5.1 Speed 52
7.5.2 Error 53
7.5.3 Distraction 54
7.6 Questionnaire Evaluation 55
7.6.1 Demographic Evaluation 55
7.6.2 No, Normal and Gradual Feedback according to User 56
7.6.3 Ranking of Feedback Types 59
7.7 User Opinions on Feedback Types 60
7.8 Observations During the Study 61

8 Discussion **63**
8.1 Acceptance or Rejection of Hypotheses 63
8.2 Limitations of the System 65

9 Future Works **67**
9.1 Improving Correlation 67
9.2 Improving Stimuli 67
9.3 Improving Feedback 67
9.4 Next Step and Future Works 68

10 Conclusion **69**

1 Introduction

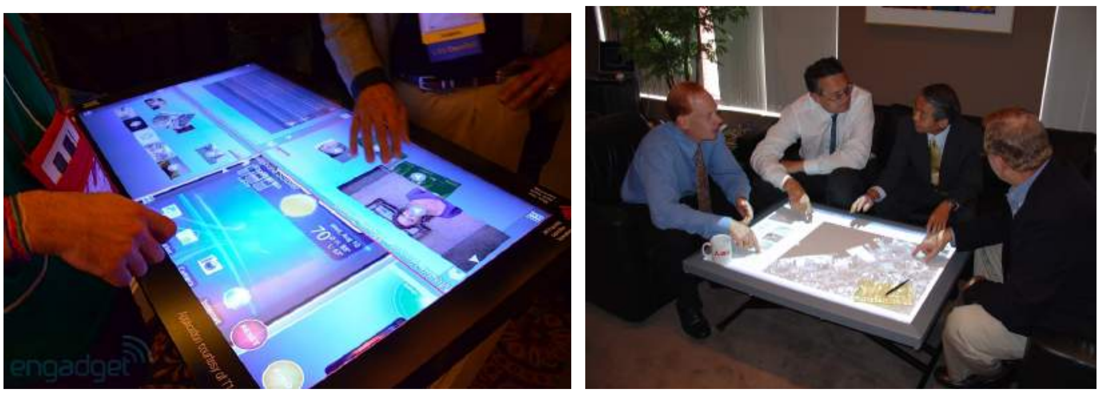


Figure 1.1: (a) a collaborative tabletop example and (b) a collaborative discussion group

Collaborative interaction on public displays is a field comprised of several of the trending areas in research on Human-Computer-Interaction during the course of the recent years. The term collaboration suggests that more than one user simultaneously utilises a computer interface at a time, whereas publicly placed displays introduce new challenges in context of the development of interaction techniques. One of these interaction methods include the use of human gaze, which has established as one of the staple HCI functionalities over the past decade.
This master thesis is aimed towards collecting all of the above mentioned aspects of Human-Computer-Interaction by researching how to display visual feedback for more than one user at a time with the given interaction tool being gaze. The problem points hereinafter are surrounding the user's need of feedback by the system for a comfortable interaction process.
Normally, during a single user desktop PC setting, the user knows precisely which part of the screen they are looking at and which object they are selecting at the moment. Even without getting visual feedback in form of on-screen pointers, indicating the selection, the use of the given interface is still efficient for the single user.
Having multiple users concurrently and collaboratively looking at the same screen as illustrated in figure 1.1, e.g. during pair-programming sessions, disturbs this efficiency, particularly if no visual feedback is given. In this case the interacting user might know about their actions but it is difficult for their partner to gauge the interaction process, as they do not know which selection is made or which part the screen is relevant at the current time, if their partner does not inform them about their intentions. This example solidifies the importance of feedback in any form. In most pair-programming sessions, typing users would naturally inform their partners about the coding intent verbally, or point at specific parts of the screen as can be seen in figure 1.2.
If we exclude the possibility to verbally exchange interaction intentions, by having the PC desktop standing in a public space where other disturbances may occur, such as high noise levels turning it difficult to understand each other, or the need to keep information confidential, feedback should take on other forms. Usually feedback is given in multiple ways conjoining visual and audible feedback. In public spaces however, the typical desktop setting yielded for larger displays which often surpass the user's height and width and thus reach. Interacting on these displays collaboratively usually includes the use of touch as the preferred interaction tool. Touch-to-select is one of the most natural ways of interacting with computer screens, as they include the human sense of touch and feel according to Pavlovic et al. [46]. Albeit this fact, sometimes public displays cannot be mounted in a way for the user to comfortably have access to the screen, e.g. if the space is not available, as the public display setting tend to become bulky as can be seen in figure 1.3, which shows an information wall inside of ION Orchard, a shopping mall in Singapore. Figure 1.3 further shows a board displaying flight information at the Hongkong airport.

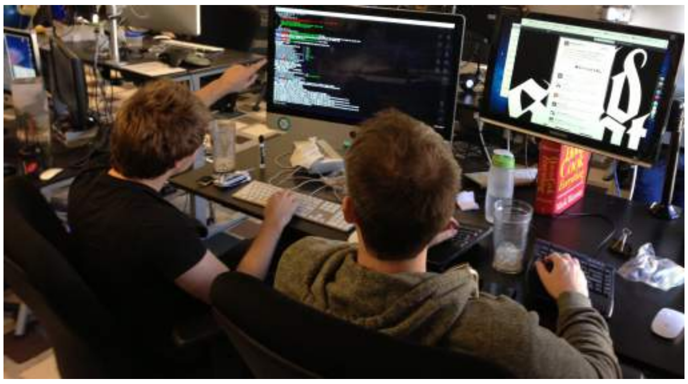


Figure 1.2: Pair-Programming as example of inter-human interaction requiring feedback

In these cases gaze emerges as the fitting tool of interaction, as it is also considered a natural way for human to interact with their surroundings, particularly computer displays. However, gaze by itself cannot make any selection, as the need for verification of the selection still remains. For that matter the use of pursuit interactions seems the most plausible to include into collaborative public display interactions.
The objective of this thesis is to assess different types of visual feedback on their usability and thus trying to provide an enhancement to collaborative working on public displays for more than one user simultaneously. The thesis is structured as follows:
After summing up the related work on collaborative and public displays, as well as gaze interaction, including their challenges in terms of incorporating multiple users in chapter 2, we will give an outline of the underlying concept as well as mention the hypotheses formulated specifically for this thesis including their null hypotheses in chapter 3. Furthermore chapter 4 will mark the beginnings of the thesis coding work to track gaze using RGB cameras. In addition to that, chapter 5 gives an explanation on the implementation of the user interface. Following this, we will present the user study settings in chapter 6, followed by the evaluation results in chapter 7. With the resulting two chapters 8 and 9, containing a discussion on the findings of the user study and the subsequently announced future challenges to approach, we will conclude the thesis by summing up all the chapters shortly and drawing a conclusion.

Figure 1.3: (a) public touch display inside of ION Orchard and (b) flight information at Hongkong airport

2 Related Work

To gain a better understanding of the background behind choosing to focus on multiple users and gaze interaction it is necessary to give information on several current trends and research areas in the field surrounding *Human-Computer-Interaction*. The following chapter gives insight into the three basic thesis building topics: HCI on public and collaborative displays or tabletops, feedback methods for gaze pursuits and gaze interaction involving multiple users.

2.1 Public and collaborative displays

One of the trending subjects in HCI aims to enable user interactions on larger-sized displays is often found in public spaces, that require techniques for immediate use at a given time, is called research on public displays. Such displays can be placed standing, e.g. mounted onto walls or lying horizontally, i.e. tabletops.
Large public displays offer a fair amount of possibilities to improve ubiquitous interaction according to Buxton et al. [24]. They usually function as information tables and can be found at many public spaces such as airports, train station, etc. As collaboration is an important factor of inter-human interaction, research also focuses on taking this everyday occurrence into HCI, therefore aiming to enable collaboration or multi-interaction on public displays [55]. Russell et al. built *BlueBoord*, a display allowing for collaborative work in small groups by distinguishing each user using an RFID reader, considering tasks such as sketching ideas, comparing notes or sharing content [55]. Their concept can be seen in figure 2.1, which shows the personalised interface for each user as well as how interaction takes form.

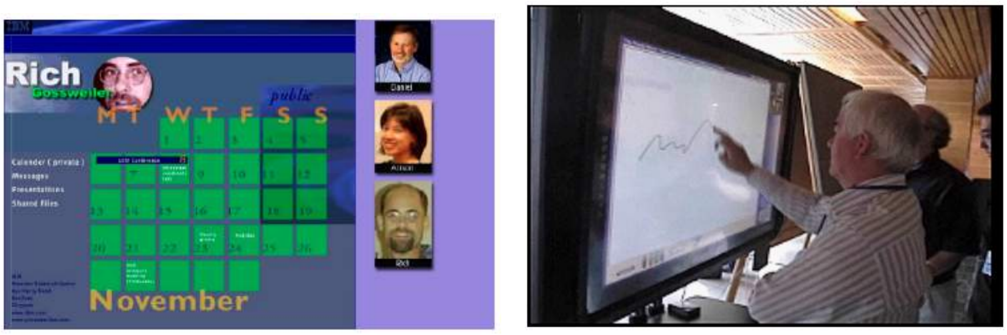


Figure 2.1: (a) BlueBoard interface design and (b) interaction on the interface

They observed common occurrences during the interaction process and listed these according to group use and single user interaction. As *BlueBoard* did not have multi-touch support, users had to learn how to collaborate, because multiple people touching on one screen resulted in the cursor jittering around instead of focusing on one point of the screen. Furthermore Russell et al. managed to list observations on emergences, such as the process of establishing a leader mentality during group interactions which describe the social aspects of collaborative interaction in general [55].
Another approach for collaborative interaction on larger displays is *MERBoard*, which was used during the Mars exploration by NASA, who deployed two rovers designed to explore Mars, while the members of the evaluating group discussed the data in a real-life collaborative setting [33]. These types of collaborative display interactions however, were observed for semi-public scenarios, where small groups with purpose and prior attention to the screen would interact together [32, 47].
Other research in this topic was conducted by Jakobsen and Hornbæk, who examined how collaboration on a larger displays worked. They had participant pairs, who each had their respective place in front of the screen and could comfortably switch between parallel and collaborative work, fulfill

tasks around browsing, navigating and researching [36]. According to researchers they also tried to find a relation between visual attention and verbal communication, resulting in an enhancement during collaboration.
On the other hand, Peltonen et al. focused on observing how user groups approach public displays in particular, i.e. starting from what caught their attention to how they used the screen and how they behaved during the interaction process [47]. They stated that their developed display, *City-Wall*, could identify specific fingers on a hand and multiple hands on the screen, as well as gestures for people ranging from child to adult age with different postures and heights. The display aimed at having multiple people interacting simultaneously on image manipulation, i.e. gallery browsing and image scaling or translating, [47].
During this study no specific types of notifications were used to attract passing people but instead the display was placed in a crowded and busy area. The researchers could distinguish different types of uses for the display, ranging from collaborative teamwork, parallel but still singular use and conflict use. These types of interaction denoted another part of research on public displays. Most often such displays would be placed in public spaces to collect mass data on a specific topic, such as schools, e.g. in research on *CWall* in Campiello, Venice which provided information on current community issues and interests [13] or Dynamo, which also allowed for private interactions by reserving space on the public screen and observing how users attracted other potential users [23].
Taking into consideration larger-sized displays in general, there are several challenges that can occur during their use, which will be explained in the following.

2.2 Challenges using Public Displays

There are several benefits paired with projecting collaborative interaction onto a public display, some of which concern the use of a larger screen in general, as described by Czerwinski et al. [28]. Working with bigger screens according to them encourages productivity, as well as periphery awareness and recognition memory improvement [29]. Especially with normal map applications and in the 3D space, having a larger-sized screen leads to better navigation overall and thus user satisfaction, according to Ball and North, as well as mentioned by Tan et al. [16, 60].
However, there are several challenges emerging when using public displays. First of all, as the term large-sized public display suggests, there are several social aspects that need to be considered both for multiple users or individual users, such as learning effects of other nearby users, how multiple people should behave while collaborating on a public display, whether there were leader tendencies, etc. [55].
Interaction on larger-sized displays comes with several basic usability issues according to Czerwinski et al. [28]: The size of the screen can be overwhelming in regard to the distance between user and screen, which results in direct influence on the reachability as well as the accessibility.
In terms of public displays, there are also several issues appearing. First of all, public displays are usually found in crowded and busy spaces. People often do not actively pay attention to their surroundings and might not spot the interactivity of the display at first glance. Thus, attracting a passerby's attention is one of the main challenges still present in research on public displays as stated by Davies et al. [30]. There have been attempts to alleviate this issue in the past, such as having interactive adjustable tasks whenever the attention of the user is on the display and the user is tracked [43].
Another challenge presented on public displays is the fact that privacy for the respective user is normally not guaranteed, as there is always a risk of other people shoulder-surfing or observing the respective user in their interaction, according to Alt et al. [14]. Attempts to solve this issue involves exchanging sensible contexts with the user's mobile devices to double-check or verify possible confidential information, instead of using the screen, as mobile devices, due to their size, provide more privacy according to Alt et al. [15].

In addition to that, working with public displays requires the user to be able to spontaneously walk up to the display and instantly start interacting without having to undergo procedures such as login beforehand as stated by Davies et al. [30]. Interaction on public displays needs to proceed in a fast manner, as it occurs coincidentally during the course of seconds, according to Müller et al. [45].
The third big challenge on public displays in terms of user interaction speaks of social aspects. Public displays can, as previously mentioned, lead to embarrassment during interaction, e.g. people have to perform certain gestures in order for the display to recognise and start tracking them, according to Bignull and Rogers [22]. Mäkelä et al. spoke of external influences which they further categorised into critera such as weather, events, surroundings, space, inhabitants and vandalism. They then tried to research projects either successfully solving these problems, acknowledging them, ignoring them, or embracing them and sorted them accordingly [44].
Also a problem is how public displays are approached. As mentioned earlier, public displays are usually larger-sized screens, which in turn renders it more difficult to deploy them in crowded spaces, often resulting in the users having to look at them from a distance instead of standing directly in front of them, as stated by Davies et al. [30]. This further impairs the reachability the user has on the display.
A method to improve usability of public displays which has been researched is the use of gaze as input for interaction. Gaze tries to solve the above mentioned challenges, such as can be seen with the work of Zhang et al., who adjusted contents of a public display to the approaching passerby by showing video feed as soon as the user was close to the display and thus recognised. Furthermore they observed how a differenct problem of needing to call other users' attention is solved because of a honeypot effect [70]. As previously stated, gaze is a modality which increasingly finds interest in terms of interaction with public displays, thus becoming more integrated into HCI research on the use of public displays.

2.3 Gaze Interaction

As mentioned, gaze interaction has evolved into one of the profound interaction methods that HCI research involves and offers. Eye movement is considered quick, effortless and very precise and thus natural to use as means of interaction as stated by Ware and Mikaelian [66].
Furthermore gaze normally is incorporated into the interaction process in any case, as users tend to look at what they are interacting with, perceive it and then continue to manipulate the interaction object according to Chatterjee et al. [26]. Most of these times, using gaze to interact with ubiquitous devices results in gaze being used as means of selection prior to manipulation as mentioned by Chatterjee et al. [26]. According to them gaze functions in addition to existing selection means such as free-space gestures, as the hands also are an important natural means of human manipulation in real-life scenarios, as stated by Conolly [27] or Wilson [67].
Other means of selection often paired with gaze involve touch gestures or pen as stylus as stated by Pfeuffer et al. [51]. In the following section we will look at the different ways gaze can be tracked, followed by a possible interaction techniques, which are important when using eye-tracking to interact with ubiquitous devices, particularly taking a look into gaze pursuit interaction and gaze as aid in selection processes. Furthermore we will assess how feedback is given for gaze-related interactions and finally discuss how multiple users are incorporated into gaze-related research in HCI.
There are several open-source as well as paid frameworks providing gaze tracking algorithms, such as *EyeSee*[6] or *xLabs*[12]. These technologies operate using different coding languages and, as of now, let the user try out how gaze tracking works using web browsers. However, they require calibration prior to usage. The providers also expect a prior registration and are therefore not openly available for anyone. Important to mention is, that these frameworks use webcam recorded gaze data for their gaze tracking, which forms one way to collect eye movement data.
With gaze interaction rendered interesting for research in HCI, researchers have been trying to

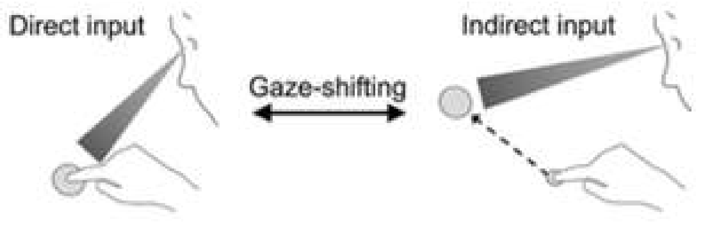


Figure 2.2: Theory behind gaze-shifting: Use gaze as focusing device or moving device

steadily improve gaze input collecting processes, by introducing several types of devices to aid in tracking the user's eye movement. The majority of said tracking devices can be acquired on a user-friendly budget [11, 7], sometimes for less than 200 Dollars per tracker package.
These types of trackers remain unintrusive as they can be mounted onto the bottom of the display. To keep the screen visible at all times and still track a decent amount of gaze data, researches also use head-mounted tracking devices such as glasses, e.g. by Shimizu et al. who implemented *Solar System*, a smooth pursuit based gaze interaction system using EOG glasses instead of trackers such as from Tobii or SMI [57].
As there are different ways to track gaze the input traits vary according to Feit et al., who stated that gaze interaction data collection and specifically data precision are dependent on the particular user, tracking device and position in regard to the screen among other criteria [31]. They further speak of ways to make gaze interaction more robust throughout the tracking methods by comparing the data collected in the same environment using two different methods of gaze tracking and found that especially data loss occurs near the edges of the screen which affects overall precision and accuracy. The researchers also provided ideas to optimize this common problem, such as trying to calibrate the tracker using more points and avoiding to place stimuli in these areas.
We now know that several ways exist to track gaze data and what problems might occur, but we will have a look at how research uses gaze to interact with ubiquitous devices in the following.

2.3.1 Gaze as means of Selection

In many research topics surrounding gaze in HCI, eye-tracking and mapping is used as tool to enhance interaction, i.e. gaze fixations act as general direction of the selection, whereas the last confirmation that a selection takes place, uses different means, such as touch or stylus devices, to avoid unintended selections, because of *Midas' Touch* [35]. This is a common problem on gaze-only devices, as the system thinks that the user is already selecting an object while they are still roaming the display for items of interest.
A lot of research has been done in terms of combining gaze with other modalities to create multimodal interaction on the same screen, such as e.g. Pfeuffer et al. who developed *Gaze-Touch*, which is based on gaze as selection tool and multi-touch as manipulation means on multi-touch supporting displays [50]. According to them gaze-touch complements direct-touch and they tried to compare the two interaction techniques and provide possible applications, such as map navigation or colour selection during drawing tasks. During this scenario, Pfeuffer et al. also introduced *Gaze-Shifting* which makes use of the above explained gaze-touch method and allows for shifting between gaze as direct input and gaze as indirect input, as can be seen in figure 2.2, which shows the process of shifting between directly manipulating an object by looking at it and using touch as manipulation tool, and moving the object by still holding the selection via touch and looking elsewhere [51]. These possibilities to include gaze as means of selection, require directly touching the display either via touch or pen stylus device. Stellmach and Dachselt however, distanced themselves from the direct approach and focused on using remote interactions instead, which means

interactions were geared towards projected displays, which in turn resulted in the user standing further away from the display (e.g. public displays) [58]. Therefore, the need for a touch-supported manipulation device was evident and provided.
In addition to gaze-shifting, albeit also not using the direct approach, Turner et al. provided *Gaze + RST*, which suggests gaze to enhance rotating, scaling and translating tasks. In this case touch acts as means of manipulation for objects whereas gaze works to change the direction of the three tasks [61]. This interaction method involved remote gaze interaction on a projected display.
All in all, the topic of gaze as means of selection equals in using gaze to support interaction techniques and has been integrated as component of gaze interaction research of today, with many researchers opting to use gaze with touch on either one-person used displays or projected larger-scale displays hinting towards public display scenarios as well, e.g. Stellmach and Dachselt [59], although they still need to have a device for direct manipulation.

2.3.2 Calibration vs. No Calibration

Calibration is an important factor concerning gaze interaction. To ensure the precision of gaze as means of interaction, most of the time eye-tracking devices need to be calibrated to the screen that acts as interaction foundation. Otherwise problems in terms of accuracy occur according to Feit et al. [31].
In public situations or spontaneous interactions, this step of calibration cannot always be achieved due to time constraint or strain of having to calibrate each approaching user respectively. We need to ask ourselves how we can still keep accuracy and precision but skip the step of having to calibrate every time a new user interacts with the screen.
Huang et al. suggest leaving the calibration process in the background of the interaction and having the system calibrate itself according to user interaction. They introduce PACE which relies on eye-analysis of webcam data and comparing the data with eye-tracker collected gaze data [34]. However, this calibration system reads prerecorded data, i.e. it is a post-calibration method instead of functioning during real-time scenarios, thus not being too suitable for instant gaze interaction.
Other approaches, such as from Pfeuffer et al. or Vidal et al. aimed to use pursuits as means of interaction to combat the calibration effort [53, 65]. As for Pfeuffer et al., they opted to use smooth pursuit as means of calibration, i.e. instead of fixations for point-calibration the whole process becomes more natural for the user and thus less straining [53].
Khamis et al. implemented text pursuing as means of calibration which works on the same principle of gaze pursuit and helps calibrate the eye-tracker to a public screen, by having the user read a text, i.e. follow a line, without them knowing they are being calibrated [41].

2.3.3 Pursuit Interaction

The field of pursuit interaction takes away the need to confirm a certain selection using other means and to avoid strenuous longtime fixations that result in interaction not being comfortable for the user. Vidal et al. introduced a technique that is based on correlating gaze movement with stimuli movement [65]. The technique is describes the concept of smooth pursuit, which is a slow interaction process and happens whenever the eyes follow a movement as stated by Barnes [19]. [65]. According to Vidal et al. pursuits help to turn gaze fully spontaneous without the need to prior calibrate the used eye-tracking device [64]. The underlying idea behind pursuits is to focus on smooth movement of the eye as can be seen in figure 2.3, rather than focusing on saccades or point-calibration based on fixations as stated by Celebi et al., which also works against strain on the eyes after a long period of use, as well as the prior mentioned *Midas' Touch* problem, which occurrs mostly during gaze-only selection tasks [25]. Celebi et al. therefore also used pursuits as means to calibrate gaze.
Pfeuffer et al. used pursuit interaction in order to calibrate the tracker with the same aim [53].

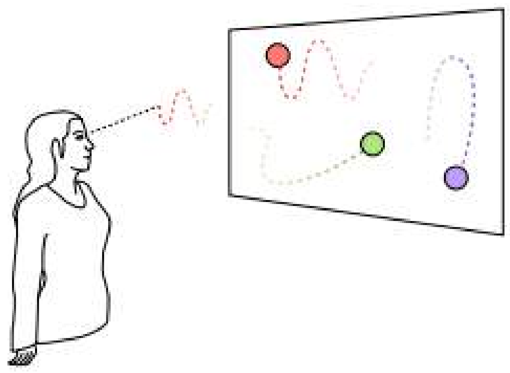


Figure 2.3: Smooth Pursuit: Matching object stimuli movement with user's gaze movement

Many pursuit interaction researchers such as Vidal et al. use the Pearson calibration coefficient to match gaze movement with stimuli movement, an algorithm which will also be used in this thesis. Because of the need for calibration prior to use is abandoned, gaze pursuit research also finds use on public displays during field studies rather than lab studies, as demonstrated by Khamis et al., who utilised pursuit interaction to significantly test stimuli moving in various trajectories on their selection speed and error rate, by implementing a game that encourages users to play [39].
Esteves et al. on the other hand made the effort to take gaze pursuit interaction away from public spaces or lab settings, to real-life interactions with smart home environment systems or daily gadgets by introducing *Orbits*, which enables smooth pursuit interaction on smart watches to control applications. Other advocates of day-to-day HCI involving gaze, are Velloso et al. who developed *AmbiGaze*, a way to incorporate smooth pursuit for the use in smart home environments [63]. They used pursuit interaction to trigger certain home technical gadgets such as audio systems, TV, etc.
If we take a look at smooth pursuit in home PC settings, Schenk et al. who introduced *GazeEverywhere* which abandonds the active use of the mouse in order to select items on the screen and focused on gaze-only interactions [56]. Albeit still sending mouse events to the computer, they projected a layer above the interface with smooth pursuit simuli.
As we can see there has been progress on smooth pursuit interaction as technique to interact or calibrate gaze, in order to help avoid persistent problems that can possibly occur using gaze-only as means of interaction.

2.3.4 Feedback

The term feedback according to Pérez-Quiñones and Silbert is an indicator for two parties communicating with each other, that both communicators have understood and acknowledged each other, thus rendering it an important factor in HCI [48]. Beun and van Eijk further stated that there are distinct types of forms such feedback can take on when a user communicates with an ubiquitous device [20]. In our case the means of communication is gaze which is an active selection tool, but does not naturally induce feedback from the other communicator. Realising feedback on interfaces supporting gaze tracking, in terms of where the system thinks the user is looking, is one of the challenges still researched in gaze interaction. Khamis et al. incorporated feedback methods for single users during gaze interaction on public displays by showing the interacting user a screen before the final selection to ask them whether they wanted to redo their selection or not [42]. The underlying method of selection was gaze pursuit and the researchers wanted to observe whether users would correct their selection if shown falsified selections and given the chance to.
Another approach to show feedback to users during gaze interaction was by providing a haptic

device and testing according to a selection task that gave users feedback whenever they looked at a specific item on the screen, according to Kangas et al. [37]. Kangas et al. further developed an approach to give feedback to gaze pursuit selections, by implementing a control tool manipulated by gaze pursuit, which had users manipulate a level bar to match the given greyscale shown above. The instant feedback occurred during the pursuit: the lower level bar would adjust to match the upper level which showed the desired greyscale as can be seen in figure 2.4. If the desired scale was met, the user could notice so immediately [38].

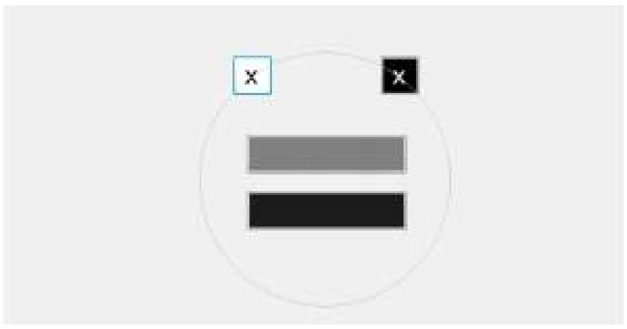


Figure 2.4: Smooth Pursuit Control: adjusting the level of the black bar to match the level of the upper bar using pursuit of the x components

2.4 Multiple User Interaction Using Gaze on Larger Displays

Many approaches to make public display interaction more interactive opt to use touch-enabling or mid-air gestures to do so, such as *WaveWindow* by Perry et al. [49], or works from Zhai et al. who combined touch-enabled wall-sized display interaction with gestures according to the distance of the user from the screen [69] as seen in figure 2.5.

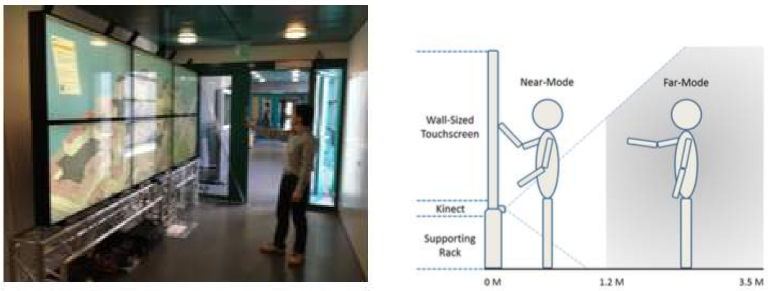


Figure 2.5: Touch and gesture combinations according to Zhai et al. [69]

In this thesis however, we will try to implement gaze pursuit for public screens, which is why we will need to focus on enabling gaze interactions for multiple users on larger-sized screens.
Oftentimes researchers trying to have multiple users interact with the same screen use tracking devices for each user such as Pfeuffer et al. [52], who built *GazeArchers*, to interactively let two users simultaneously play a game using gaze and touch on a tabletop display. The contributions they made involved testing the gaze-awareness of multiple users and have the system react accordingly, i.e. the soldiers reacting to the gaze of the user, all in the premise of a multi-player game with novel interaction technique. Therefore we can take notice that leisure activities can be built to test multiple user gaze interactions.
Another approach in involving more users on the same screen while still studying gaze awareness of the users was made by Pfeuffer et al. by having a collaborative display of information which was working using gaze interactions [54]. In this example both users are navigating with a map

and are auto-calibrated by having to follow a moving bus prior to use. The map later collapses additional information on places according to the users' interest, i.e. gaze position and behaviour. The second gaze evaluation approach having multiple users interact with a screen, is therefore based on information giving.
Gaze on public displays is mainly introduced to solve existent limitations which cannot be overcome using only touch or gestures according to Khamis et al. [40]. They further provide approaches to solve challenges common in interaction with public displays.
Zhang et al. developed a system, which takes gaze to enhance multi-user interaction via the use of map navigating tasks [71]. In this scenario two users sit beside each other each with an eye-tracking device in front of them, calibrated onto a display showing a map application. Zhang et al. try to observe how to give gaze clues, i.e. to indicate where each user is looking. Prior to that second study they researched different types of hinting gaze for users as can be seen in figure 2.6. They found that gaze hinting proved to be useful during co-located search tasks where users started dividing spaces into areas to search through. On the other hand they found a new challenge arising when giving feedback to users: there is a switch between appreciation for visible gaze positioning and being distracted by said indicators. Also, gaze in this case is very dependent on the accuracy of the tracker.
Zhang et al. give way to the direction this thesis takes on by providing feedback for multi-user interactions using gaze on larger-sized displays and show that there have been approaches in this area, which could be beneficial for further use if the accuracy problem for gaze-tracking was overcome.

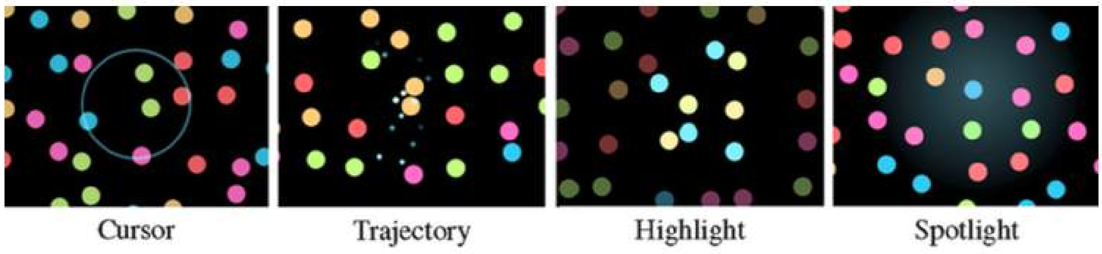


Figure 2.6: Possibilities to hint at gaze position of user

3 Concept Development

With a slight inkling as to what specific research area in HCI the thesis should take on, we brainstormed possible features to focus on. Having gained experience and interest in gaze interaction during a semester abroad, the author wanted to advance in this direction for the master thesis. We listed some ideas which were considered current trends in the research field surrounding gaze interaction.
Figure 3.1 shows the accumulation of keywords relevant for the thesis concept. The figure further puts emphasis on gaze interaction as the central topic, but seeing as this area covers a variety of directions the thesis could take on, we had to circle around current challenges and fields of attention in attempt to find the best fit.

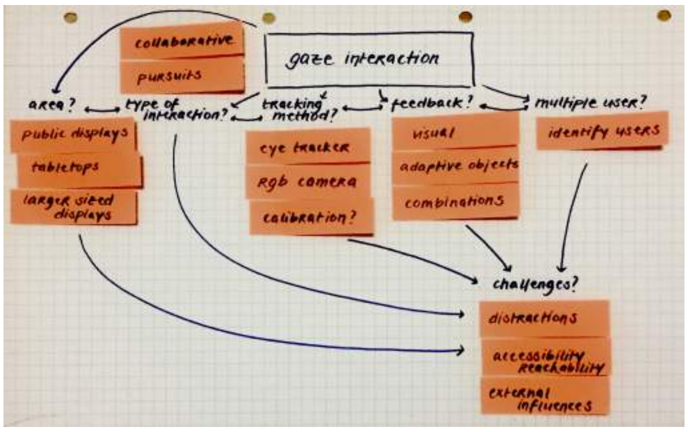


Figure 3.1: Keyword Chart illustrating the initial Concept Idea

First of all, we determined how to enable basic aspects concerning the use of gaze for the thesis, mainly regarding how gaze should be tracked during the interaction process. Research on eye-tracking as of today has a variety of possibilities to record gaze data, ranging from wearable devices such as glasses, tracking devices attached to the screen itself, e.g. eye-trackers, through to simply evaluating video feed based on the user's gaze behaviour, as mentioned in the previous chapter on background knowledge and related work.
Since the topic of multiple users collaborating on computer interfaces is a trending subject, we wanted to research on this aspect in the thesis as well. Collaborative interaction using computers is one of the reasons research groups have been swerving from the traditional desktop PC setting and therefore, looking for alternative display environments. The catchwords *public* and *tabletop* display among others are indicatives for the advancement leaning away from single-user desktop environments and towards more interactive multi-user settings.
Keeping in mind multiple user and public large-sized displays in regards to developing the thesis concept, two important questions emerged during the decision process: Firstly, how does the computer identify multiple users as such, and secondly, how should gaze be incorporated into interacting on the given display sizes. We tried to incorporate these challenges into the final research question.
Finally, we asked ourselves, what type of feedback could be given to the user based on all the above mentioned aspects, further taking into consideration how to distinguish multiple users. The following chapter will give an in-depth look on the thought process behind the final research question and the resulting concept development.

3.1 First Decision: Gaze Tracking Method

As research on mapping gaze data onto the screen using tracking devices has been established to some degree during the past years, with a majority of studies focusing on wearable and attachable tracking devices, the initial thought was to move away from the need to utilise these gadgets and instead, use an camera such as a webcam, which can be found integrated in almost all laptop devices nowadays.
This decision was made particularly to facilitate the access to eye-tracking and thus making it easier for non-expert users to try out gaze interaction without the necessity to buy specific tracking devices. Furthermore stepping away from having to possess particular electronic tools for eye-tracking and thus gaze interaction, would result in rendering this specific type of HCI more commonly known and usable for all types of expertise.
For the master thesis we have been recommended to use the OpenFace/CLM-Framework written by Baltrusaitis et al. [18]. Chapter four will explain in detail what this framework entails in terms of gaze tracking and mapping and how we employed it for our purposes.

3.2 Second Decision: Incorporating Gaze as Interaction Method

Subsequently to agreeing on using an RGB camera for gaze tracking, we determined how to incorporate gaze itself as means of interaction into the interface. As previously mentioned in chapter two, there are several ways to use eye-tracking in order to enhance and simplify user interaction with a computer interface, one being gaze as means of selection.
Since the thesis was aimed at the later use of public displays or tabletop screens, relying on this particular interaction method where the eyes act as a cursor or pointer, would presume that users had to calibrate their eye movement according to the particular screen they were interacting with. This task can become tedious especially in public environments as it demands users to maintain a certain distance from the screen at any time during the interaction process. Having this requirement would take away the benefits of public displays where users should be able to promptly start interacting with the contents displayed on the screen.
Therefore, we agreed on realising pursuits using gaze as the main method of interaction for the thesis, as it does not require tracking devices to be calibrated according to users and thus enable spontaneous interaction with a given screen at any time.

3.3 Third Decision: Type of Feedback to give to Users

For a long time we were unsure how to finalise the thesis concept by providing feedback according to the user. Any type of feedback on which item on the screen a certain user is interacting with becomes necessary as soon as more than one person is involved in the interaction process and pointing devices are not available.
This was the case in the master thesis, where multiple people were targeted to cooperatively interact on a display with no means to directly point at interactively used objects, i.e. not taking into consideration verbally given feedback, such as one user telling the others which objects they were looking at. Basically we were balancing between either adapting the movement behaviour of pursued objects on the screen (in trajectory or speed), or giving visual feedback according to users' gaze pursuits.
The type of feedback given should further help other users to distinguish which object a certain partner was directly pursuing, which would often facilitate cooperative work on the same screen for multiple users.
For the thesis the focus was finally shifted towards giving users multiple visual feedback types and gauging whether or not they appreciated getting them as well as to potentially see whether or not getting visual signs of selection distracted other users from their own interaction or not.

3.4 Research Problem, Hypotheses and Methods

With all prior listed criteria taken into consideration, we decided that the thesis would focus on researching gaze pursuit interaction for multiple users and analyse the opinions based on having more than one user simultaneously pursuing objects on a screen. Furthermore we would include methods to incorporate visual feedback for said users and evaluate how feedback would be perceived in such cases and if one feedback type would result in a faster or more pleasant interaction according to users. Another point to look into was whether the given feedback to one user would distract others in terms of having a cross-interaction available, i.e. both users interact with objects that are not directly in front of them but rather in their partner's interaction sphere as opposed to what happens when users interact with the same object or objects in their immediate sight area, as can be seen in figure 3.2.

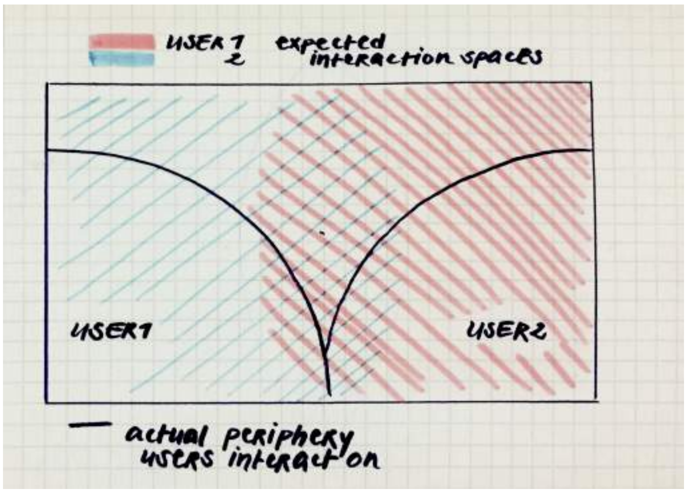

USER 1
2
expected interaction spaces
USER1
USER2
actual periphery users interaction

Figure 3.2: Interface Screen divided by Interaction Areas

We worded several hypotheses to later rectify or agree with, the entirety of which covers all the aspects we wanted to inspect further.
Firstly there are two hypotheses and their according null hypotheses on the aspect of receiving visual feedback during interaction:

Hypothesis (H1): *Getting visual feedback leads to more distractions or errors amongst users than receiving no visual feedback.*

Null Hypothesis (H1): *Getting visual feedback does not lead to more distractions or errors amongst users than receiving no visual feedback.*

Hypothesis (H2): *Getting normal visual feedback results in objects being selected faster by users than having gradual display of feedback on the screen.*

Null Hypothesis (H2): *Getting normal visual feedback does not result in objects being selected faster by users than having gradual display of feedback on the screen.*

Secondly, we formulated four hypotheses and their respective nulls on the process of multiple users interacting with spatially differing objects on the screen. For this matter, we assume that object pursuit can only happen in three types: looking at the objects closer to oneself, pursuing the same objects and gazing at the objects closer to the interaction partner.

Hypothesis (H3): *Interactions with the same objects deliver a faster selection time than having to select objects closer to the respective partners.*

Null Hypothesis (H3): *Interactions with the same objects do not deliver a faster selection time than having to select objects closer to the respective partners.*

Hypothesis (H4): *Interactions with respective objects, i.e. the objects closer to users result in a faster selection time than selecting objects closer to the partners.*

Null Hypothesis (H4): *Interactions with respective objects, i.e. the objects closer to users do not result in a faster selection time than selecting objects closer to the partners.*

Hypothesis (H5): *Interactions with the same objects result in less distractions or errors than having to select objects closer to the respective partners or users.*

Null Hypothesis (H5): *Interactions with the same objects do not result in less distractions or errors than having to select objects closer to the respective partners or users.*

Hypothesis (H6): *Interactions with respective objects, i.e. the objects closer to users result in less distractions or errors than selecting objects closer to the partners.*

Null Hypothesis (H6): *Interactions with respective objects, i.e. the objects closer to users do not result in less distractions or errors than selecting objects closer to the partners.*

The research methods utilised in order to be able to answer the given question circle around getting qualitative as well as quantitative data that can sufficiently give information on the things we want to achieve. Therefore we should aim for a case study which involves multiple users interacting together on a screen which in return gives feedback visually to react to gaze pursuit selections. The study would therefore include a survey giving quantitative answers to interaction tasks as well as interview questions to record participants' opinions. The goal of evaluation resulting thereafter is, to do statistical significance tests to either reject or accept the hypotheses and to give predictions accordingly.

3.5 Real-Life Scenario based on the Concept

To better visualise which aspects the developed concept needs to cover of we will try to explain a possible real life scenario in the subsequent part with the help of Alice and Bob, two characters created for the sole purpose of explaining the scenario. Imagine Alice and Bob finding themselves in a public space surrounded by strangers, e.g. at a foreign airport trying to find their transit gate for their connecting flight *LMU1234*. Bob notices an information screen displayed somewhere in that public area and together he and Alice manoeuvre towards said screen. In front of the screen is a stand containing the tracking devices.

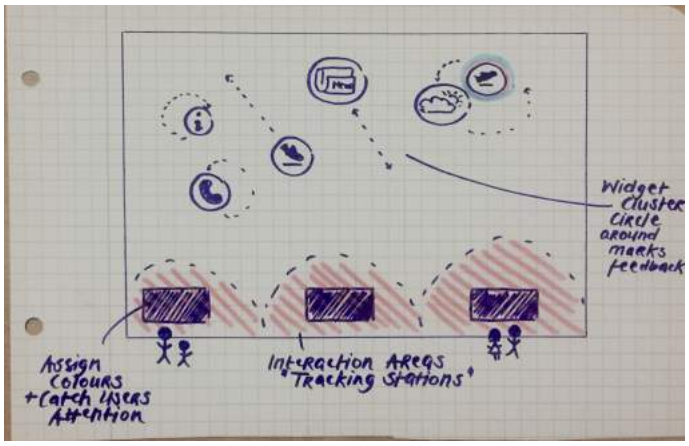

Widget Cluster Circle around marks feedback
Assign Colours +Catch users Attention
Interaction Areas "Tracking Stations"

Figure 3.3: Design of the interface Alice and Bob interact with

The next figure 3.3 shows the continuation: On the screen the both of them recognise a cluster of widgets, each of which is surrounded by a moving circle. An indicator close to where the two main characters are standing tells them to follow the circle of the widget they want to receive further details on with their eyes. The system recognises the amount of users actively interacting and assigns colours to both of them.

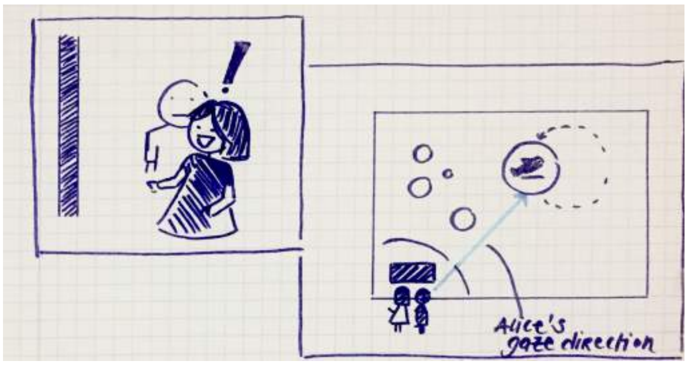


Figure 3.4: Alice spots and starts selecting the necessary widget: departure flights

In this scenario as seen on figure 3.4 Alice sees the departing flights widget, and wants to point it out to Bob, but as normal for a public space the noise level is relatively increased and thus making it harder to communicate verbally. Another disadvantage for directly pointing out the widget of interest is the size and position of the screen, which turns it harder to reach all of the display space. Furthermore Alice's and Bob's hands might be occupied holding luggage or food.

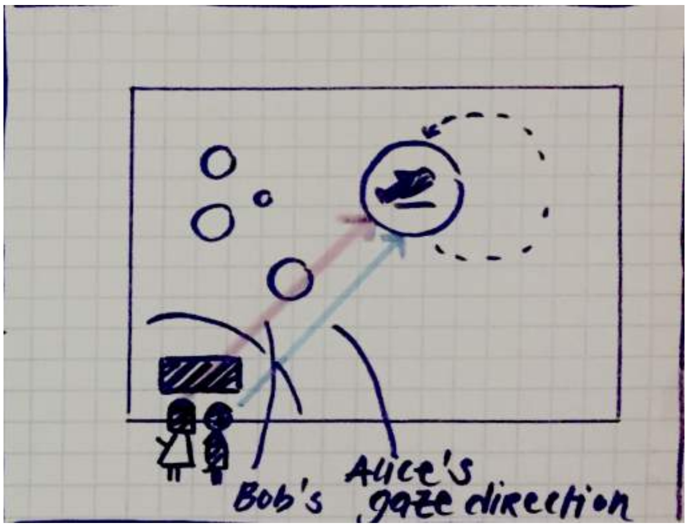


Figure 3.5: Bob joins in on the selection process

In this case as both users know their respective colours, Alice only needs to point out the rough direction she is looking at. Figure 3.5 illustrates that the system is programmed to recognise gaze patterns and match them with object movements to find correlation values. Feedback given back to Alice and Bob in visual form as it works efficiently even in places with a higher noise level and does not distract or disrupt other people nearby.

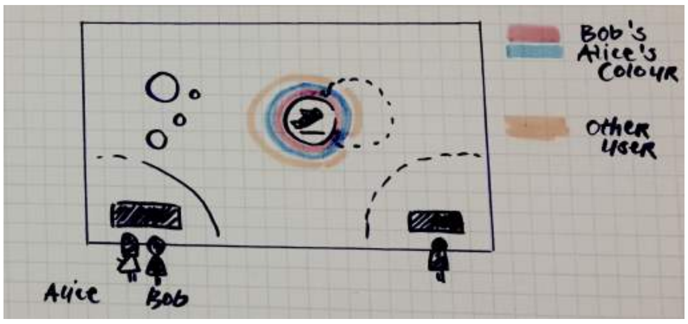


Figure 3.6: Both users receive visual feedback on their selection: The widget containing departure flights appears collapsed on their respective part of the display

Figure 3.6 shows that Bob follows Alice's gaze pattern by pursuing the coloured moving circle with his eyes as well. If the system recognises a match between Bob's eye movement and the object movement, it counts as the majority of interacting people are interested in that particular information and collapses the widget accordingly showing time slots for departing flights sorted by hour. Alice and Bob find their flight number and their gate and depart to their vacation as seen in figure 3.7.

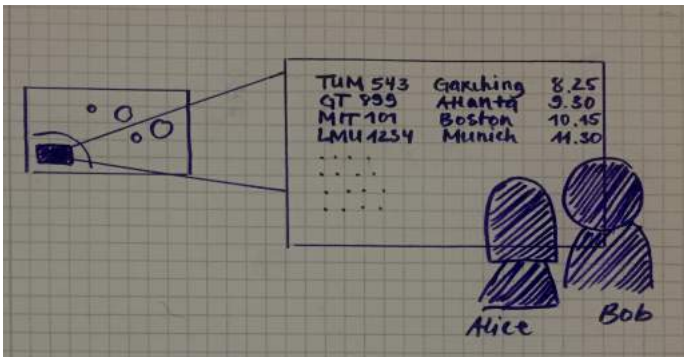


Figure 3.7: Interface Screen showing the collapsed flights

4 Tracking Gaze with an RGB Camera

During the beginning of thesis work, the proposition was to utilise an RGB camera to record users and simultaneously track their eye-movements, thus to outline their gaze behaviour. The use of common everyday cameras would dismiss the need for specific eye-tracking devices, e.g. Tobii, SMI, hence turning the prospect of incorporating gaze tracking into HCI more affordable and easily accessible.
The framework which adapted this idea was the recommended CLM-Framework (later version: Openface) written by Baltrusaitis et al., in order to provide a face analysis toolkit, which was available for public use [18]. This framework is based on face tracking and feature extraction from received video footage. Ideally, the framework also offers the extraction of eye movement, and hence gaze tracking, as stated on their wiki page. There is either the possibility to upload pre-recorded videos and track gaze according to the footage, or take live camera input. For development purposes we chose to use the feed provided by the attached webcam camera, which theoretically could be swapped for any video recording camera at hand.
Prior to developing and coding, we set up the framework with the use of the required Visual Studio version, which is stated on the CLM github page. The code for the framework was written using C++, an objective programming language similar to Java or C.
As for the build of the code structure, the framework contained several built-in programs, which made use of the given features and aimed to showcase the different extraction methods announced on the website. we chose to further work with the feature extraction project, which returned the given video input with the respective facial features outlined using dots. Furthermore, the gaze direction was also illustrated as can be seen in figure 4.1, with a green line protruding from the pupil to the estimated point on the screen.

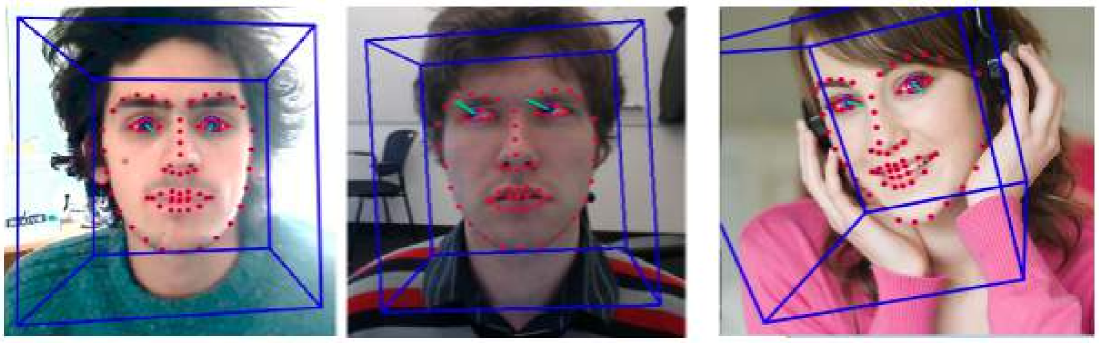


Figure 4.1: Feature extraction including gaze vector using CLM-Framework as taken from the website [3, 9]

According to Wood et al., who included gaze estimation within the framework, they developed a dynamic eye-region model which renders realistic images around the eye area using 3D head scans using a simplistic eye-model and what according to their terminus the effect of *retopolizing* the face geometry, to create a more realistic image while also allowing to extract features according to head positions. Interesting was the fact that this novel model allowed for eye-shape registration and iris tracking based on the use of webcams [68]. For that matter they trained a Constrained Local Neural Field (CLNF) deformable model, which was trained with landmark location data generated by their method [17]. The authors further stated that they chose to not focus on pupil tracking as it was hard for them to distinguish pupils from most of the images they received from a subset of MPIIGaze. MPIIGaze is a dataset containing face images and screen-situated gaze locations which were accumulated previously. Figure 4.2 shows the recognitions of eye shape and iris illustrated on in the wild accumulated tracked images on the one hand and webcam images on

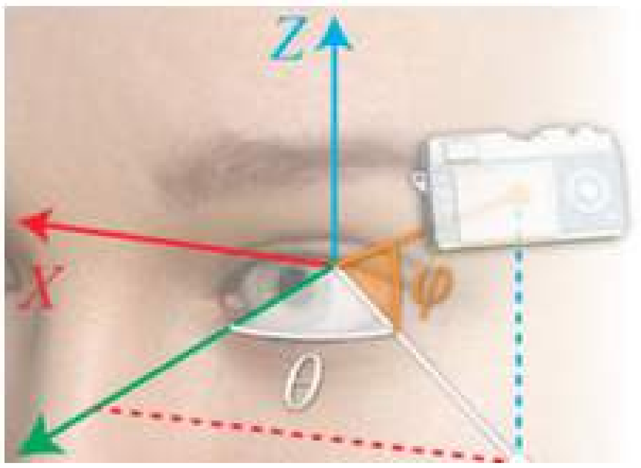


Figure 4.3: Gaze estimation eyeball model with camera positioned dynamically changing in regard to pose

the other. The image size in this case does not matter as the system successfull recognises facial features, as long as there is no occlusion of the eyes by foreign objects present.

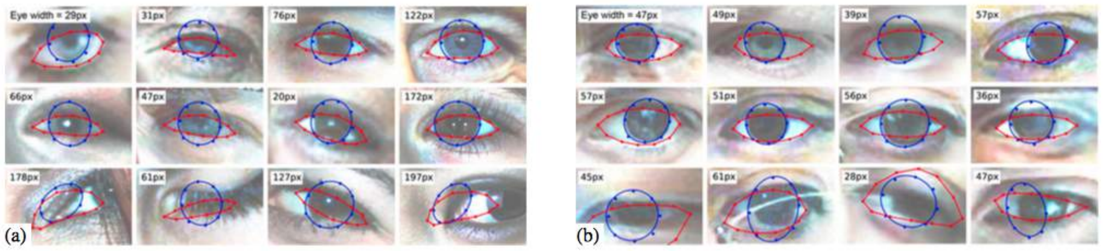


Figure 4.2: Eye and iris extractions for (a) in the wild images and (b) webcam images. The first two rows illustrate correct registrations regardless of image resolution, whereas the last row illustrates failed cases which were caused by occlusions or closed eyes, etc. [68]

The following sections summarise attempts using the framework in order to realise pursuit interaction and track multiple users simultaneously. Since the website stated that extracting features for multiple users was a possible functionality, we opted to start with printing out gaze data and visualising the gaze points onto the screen in a first approach.

4.1 First Attempt: Obtaining Gaze

The framework comes equipped with a class GazeEstimation, which offers the functionality needed to extract eye features as well as illustrate the general gaze direction. The method estimates gaze provided by finding the pupil position as a 3D point according to facial feature extraction. This feature extraction process uses the implemented model and calculates a vector by dividing said point by its normalised form. The newly gained 3D point is called ray direction and is taken as a vector to intersect with the calculated eyeball centre to receive the gaze vector using the gaze estimation eyeball model in figure 4.3. The model positions the camera relative to the head and eye position. The camera position is at Point3f(0,0,0), the zero point, thus marking the start of a 3D coordinate system. According to log printouts there is a coordinate system for each eye which renders it more complex to map the gaze onto the screen. Although the visualisation implemented by CLM itself hinted the general direction of the gaze vector on the screen, obtaining actual gaze

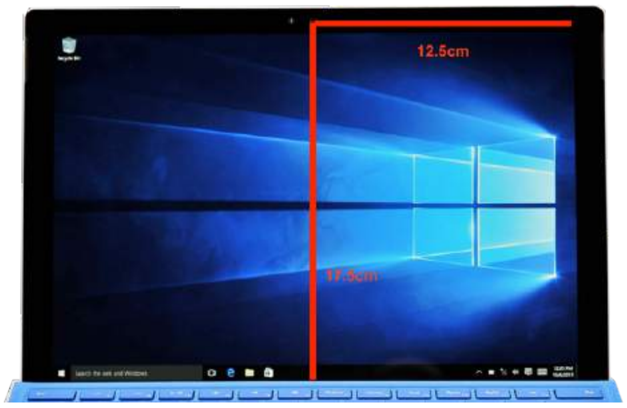


Figure 4.4: Measurements of the display from middle of the webcam lens to respective outline of the actually used screen

data resulted in values of absolute nature, i.e. gaze points achieved were 3D points rather than 2D points. We needed to map them, for them to become screen coordinates. The first step we tried was to measure the distance between the webcam lens and the outer corners of the screen, since the camera was as previously stated positioned at the zero point. We initially started developing using a Microsoft Surface Pro 4, therefore the values we measured were approximately 12.5cm from the middle of the camera lens to the respective corners left and right, and about 17.5cm from the middle of the camera straight to the bottom line of the screen which is illustrated in 4.4. Notice, that this is measured for the display part which can be interacted with. We calculated the screen coordinates according to the gaze vector by using the absolute screen length and width, as well as the measured values, to map the estimated 3D point to a 2D pixel on the screen, by taking the value received, when looking into the corners of the screen, and dividing them accordingly to the measured values. The results showed that the calculated coordinates did not match the actual gaze positions when tested, so this approach was not considered sufficient for illustrating gaze points on the screen.
The second approach involved projecting 3D gaze points on the screen as 2D points. In this case, focal length and optical axis were mentioned as two requirements in order to calculate the projected gaze point. These specifications however, were not provided by Microsoft as taken from the specification list for the Surface Pro 4. After contacting the author, we learned that in order to receive the correct screen coordinates we would have to map the gaze points using the calculated screen coordinates themselves, which would result in needing sample sizes of users looking at different corners of the screen to estimate the best transfer of 3D to 2D coordinates. In addition to that, adding information on focal length, optical axis and how far the user was situated as compared to the camera position was necessary, which is illustrated in figure 4.5. Since these calculations seemed to be a fairly complex topic in themselves, we decided to abandon the attempt to project the 3D gaze points onto a specific screen altogether and try out direct correlation. Gaze pursuit tries to correlate gaze movement with object movement.

4.2 Second Attempt: Correlation

Rather than projecting the 3D gaze point onto the screen, the second approach involved trying to correlate gaze with object movements. For that matter, we implemented a simple interface with two circles moving around using different trajectories, one linear and the other one circular. This was implemented using SFML, a multimedia library which facilitates drawing and animating as well as being useful for threading purposes. In this case, object coordinates were saved in a list whenever gaze movement was detected which was saved in a separate list as well. In this case we ignored the z-coordinate of the gaze vector, as the movement occurred in a two-dimensional

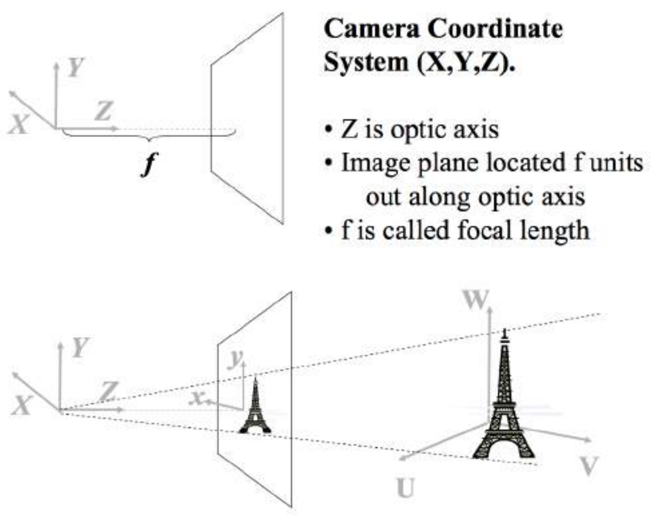

Camera Coordinate System (X,Y,Z).
• Z is optic axis
• Image plane located f units out along optic axis
• f is called focal length
X
Y
Z
f
x
y
W
U
V

Figure 4.5: Projecting 3D points onto a 2D plane: Necessary information on (a) camera positioning and (b) concept of projection step taken from Penn State University's lecture on Computer Vision [5]

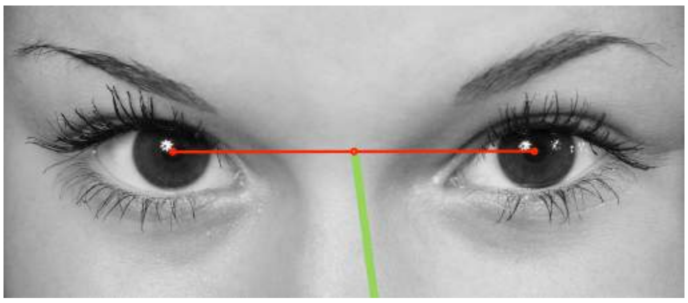


Figure 4.6: Intersecting gaze to find a generalisation and mid-point between pupils

manner.
Every 500ms we calculated the correlation values for linear and circular movement with the given samples, and printed them to see if one or the other would result in a higher correlation value. In this case however, there was no significant difference to observe between the printed correlation values for one eye. Since the right eye and the left eye were calculated separately, the correlation values had to be calculated separately as well. The hereinafter problem was that in doing so, taking the average of the calculated values in order to receive overall correlation would result in a smaller correlation value, hence we needed to make the gaze itself more consistent.

4.3 Third Attempt: Mid-Pupil-Point

To overcome the problem of correlation value averages being very low, we decided to bring both the gaze vector of the left eye and the right eye to a middle point first, by calculating the mid-point between both pupils and estimating gaze based on this vector point. In order to receive the mid-point vector, first of all, gaze vectors for both eyes were found using the given gaze estimation method. The resulting vectors were extended and the intersection between those two calculated, as illustrated in figure 4.6 Taking the generated vector we proceeded to continue calculating correlation values according to the steps mentioned in the second approach while using the sample regardless of its size after every 500ms, to generate the correlation values. Still, the results differed only minimally and we also started to experience signs of lagging movements of both gaze and objects, albeit object movement being displayed using a separate thread. we concluded that this approach would not help me achieve our objectives either.

4.4 Final Attempt: Post-Process Pursuit

For this approach we took the above mentioned logging of gaze and object movements and saved them in a file, containing the following information: timestamp, linearX, linearY, circularX, circularY, gazeX, gazeY. The respective x- and y-coordinates were logged in addition to the timestamp at which they appeared, i.e. were detected.
Prior to the correlation calculations, as they were processed post-time for this approach, we logged the interaction by recording the video feed while simultaneously logging the object movement data using the same framerate of 30Hz. The task was to pursue the circles alternating for 5s at each time. Afterwards we cut the video at start and endpoint of the interaction and processed it using the CLM-framework to log gaze detected by the system separately.
We imported the results of our interaction into Google Sheets and used the correlation function provided to calculate correlation values. We did so for a sample size of 10 and 20. Having the task sheet available, we matched whether the calculated correlation value matched the correct circle selection. The results unfortunately, albeit correlation values being higher than during real-time processing, did not accurately point out the right object for all times.

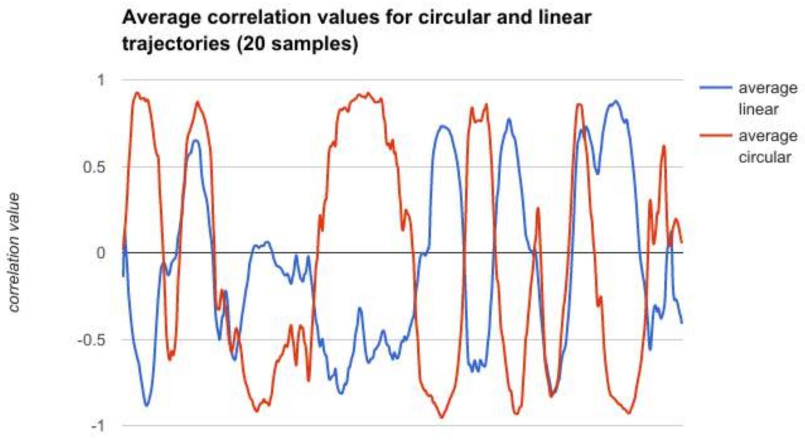
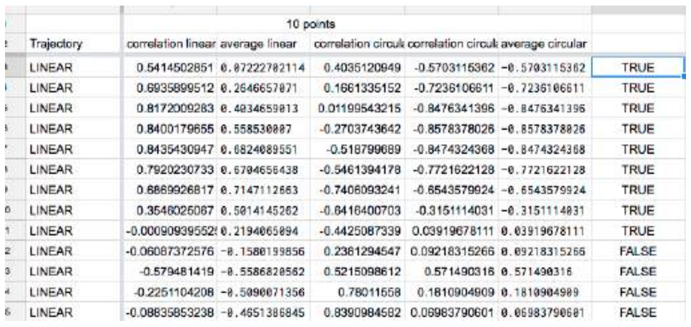


Figure 4.7: Comparison of average correlation values for post-processed gaze and object movement on a sample size of 20

Figure 4.7 shows that the comparison between the average correlation values were different from each other given the task of object selection, and the chart further shows that most times either one or the other of the two trajectories provided a higher value, hinting that only one object would at a time be marked as being selected, whereas the correlation value for the other object was not high enough. But having a closer look at the matching between expected trajectory and the object being looked at, we can see in figure 4.8 that the matching did not always confirm that the right object was looked at during a specific time. Because of that observation, this approach was abandoned, since it did not provide the satisfactory results in distinguishing the right selection. Notice, that our approach in pursuit calculation uses the pupil position and movement to correlate with object movement, as opposed to previously stated research such as from Vidal et al., that utilised uncalibrated gaze movement mapped onto the screen to match with stimuli movement [65]. However, the calculations did provide a higher correlation value in comparison to real-time



	10 points					
Trajectory	correlation linear	average linear	correlation circul	correlation circul	average circular	
LINEAR	0.5414502851	0.07222702114	0.4035120949	-0.5703115362	-0.5703115362	TRUE
LINEAR	0.6935899512	0.2646657071	0.1661335152	-0.7236106611	-0.7236106611	TRUE
LINEAR	0.8172009283	0.4034659013	0.01199543215	-0.8476341396	-0.8476341396	TRUE
LINEAR	0.8400179655	0.558530007	-0.2703743642	-0.8578378026	-0.8578378026	TRUE
LINEAR	0.8435430947	0.6824089551	-0.518799689	-0.8474324368	-0.8474324368	TRUE
LINEAR	0.7920230733	0.6704656438	-0.5461394178	-0.7721622128	-0.7721622128	TRUE
LINEAR	0.6869926817	0.7147112663	-0.7406093241	-0.6543579924	-0.6543579924	TRUE
LINEAR	0.3546025067	0.5014145262	-0.6416400703	-0.3151114031	-0.3151114031	TRUE
LINEAR	-0.00090939552!	0.2194065094	-0.4425087339	0.03919678111	0.03919678111	TRUE
LINEAR	-0.06087372576	-0.1580199856	0.2381294547	0.09218315266	0.09218315266	FALSE
LINEAR	-0.579481419	-0.5586828562	0.5215098612	0.571490316	0.571490316	FALSE
LINEAR	-0.2251104208	-0.5090071356	0.78011558	0.1810904909	0.1810904909	FALSE
LINEAR	-0.08835853238	-0.4651386845	0.8390984582	0.06983790601	0.06983790601	FALSE

Figure 4.8: Trying to distinguish a match between expected and actual trajectory. FALSE marking that the object with the higher correlation value was not the expected object and vice versa

pursuit calculations. But it also had a major problem: The question of how to give feedback in real-time to the user, if their interaction would be post-processed, remained unsolved and became a challenge that would result in further research being needed.

4.5 Summary

After attempting to use first of all approaches to map gaze vectors and position onto the screen as well as trying to directly implement pursuits using the pupil position instead of the gaze point on the screen, both in real-time and post-processed, we concluded that in order to improve the pursuit correlation calculations we would need either a higher memory storage to keep latency times low or spend more time on CLM and its features in general. The first approach we made using the received gaze vectors and mapping them onto the screen did not provide sufficient results, because we would have needed to map the calculated gaze based on the projection of the gaze data onto a XY plane overlay at the camera position in order to calibrate the gaze according to the specific camera and further provide information on the focal length and optical axis of the camera in use, as well as the distance between camera and user. This would lead to users having to maintain an unnatural and uncomfortable position in front of the screen, which did not enhance usability at this point.
The second approach paired with the third approach did give me results in terms of correlating object and gaze movements, however, the results were not satisfying in regard to highly differing correlation values, i.e. none of the objects was noticeably marked as selected because of a clearly higher correlation value. This was caused by lagging during the real-time processing part, as the CLM-framework and its functionality seemed to slow down considerably on the Surface Pro, as well as on an ASUS desktop PC we tried running the code on. Latency and lagging resulted in low correlation values for both the tested moving objects, which did not help in strengthening the advantages for the use considering gaze pursuits.
The conclusion to that led to a final approach using pre-recorded video footage of an interaction. This approach, although seemingly sensible to reduce latency and lagging, registered two problems: first of all, there was no clear way to give the user feedback during their interaction if the interaction had to be pre-recorded first and processed at post-time. Secondly, the results delivered higher correlation values but often it was indicated that the wrong object was selected by the user. So this approach was abandoned as well.
Because of these attempts, we came to the conclusion that the use of RGB cameras with the CLM-framework was not a promising way to enable pursuit interaction and thus suit our purposes. It would need more research on optimising gaze mapping or pursuit calculations if we intended to further use CLM. Therefore, we decided to abandon the use of RGB cameras in favour of utilising eye-trackers for further development.

5 Implementation

After having made the decision to further rely on Tobii eye-trackers to enable gaze interaction for multiple potential users, the focus pointed to developing the interface displaying the different visual feedback methods.
We opted to continue all tasks considering software development on a laptop running the Windows OS as it supports the Tobii SDK. Furthermore all coding was done using Java as the main programming language, as it offers a variety of libraries and frameworks for graphical interface design, as well as a well-described API for multithreading purposes.
For that matter the following chapter describing the implementation process will be divided into two main parts: Firstly, a listing of helpful frameworks and libraries applied, as well as what they entail and offer in terms of being useful for the application to be developed.
And secondly, an implementation procedure based on step-by-step milestones. This part will also give an insight into important algorithms and classes as well as explain the idea behind choosing certain coding actions.

5.1 List of Frameworks and Libraries

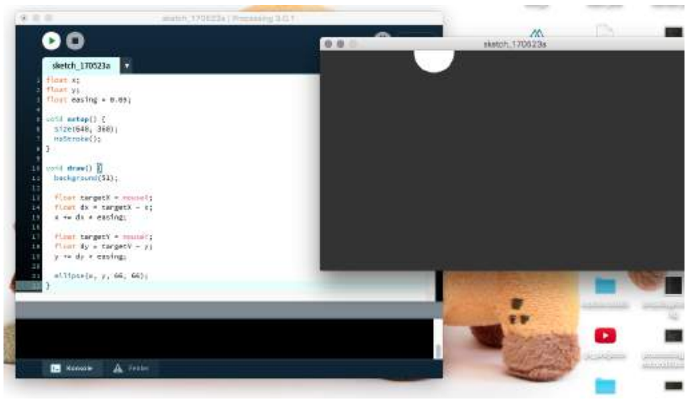
With Java being the coding language of choice, and having abandoned the further use of the CLM-/OpenFace-framework for gaze tracking, the hereafter mentioned frameworks and libraries have been found useful to design the interface, as they facilitated the coding process and thus helped in achieving the result faster.

5.1.1 tobiisdk4j - A Wrapper to use the Tobii SDK with Java

This wrapper written by Ralf Biedert successfully binds the latest Tobii SDK to Java and allows me to obtain gaze data by simply calling a function [21]. Unfortunately, as of date the code available on GitHub has been taken down by the owner himself. The wrapper is referenced in the Java project, so that we do not have to be concerned about mapping gaze anymore. Having used the wrapper in a project prior to the thesis, we also had the advantage of experience using the code already which made choosing to use tobiisdk4j easier.
Another plus was that the wrapper allowed to connect multiple eye-trackers simultaneously to a system and was already successfully tested for the Tobii REX trackers, which we made use of during our user study. Also important was that the latency was kept at a minimum to not slow down the interaction process, which is provided by tobiisdk4j, as it keeps the latency below 1ms, as stated by the author.
It is, however, necessary to mention that the wrapper code only works on 64bit JVMs. The needed version can be downloaded, thus this disadvantage can be overcome easily.

5.1.2 Processing 3.0 - A Visual Arts Context Programming Language

Processing is considered an interesting open source coding language which can easily be integrated into Java. It allows for simple and fast interface prototyping, as well as more refined graphics works, which can be found displayed on the website.
Furthermore the API is thoroughly explained and has visual examples accompanying the code samples. For the purpose of our interface we needed a possibility to draw circles and animate them to move in different directions and trajectories. Furthermore we wanted to have the ability to change colours simply with one line of code, which is provided by Processing. The programming language also has the possibility to add sounds at specific times. Figure 5.1 shows the interface that allows for fast prototyping using only a few lines of code [10].



```
float x;
float y;
float easing = 0.05;

void setup() {
  size(640, 360);
  noStroke();
}

void draw() {
  background(51);

  float targetX = mouseX;
  float dx = targetX - x;
  x += dx * easing;

  float targetY = mouseY;
  float dy = targetY - y;
  y += dy * easing;

  ellipse(x, y, 66, 66);
}
```

Figure 5.1: Processing interface and instant visualisation

5.1.3 Commons Math - The Apache Commons Mathematics Library

We chose to use the library solely for mathematical purposes, as it provides pre-defined mathematical functions and constants, especially in terms of calculations regarding correlation values [4]. The library offers a class PearsonsCorrelation which according to the API uses pairs of data arrays and computes the correlation coefficients.

5.1.4 Minim - An Audio Library for Processing

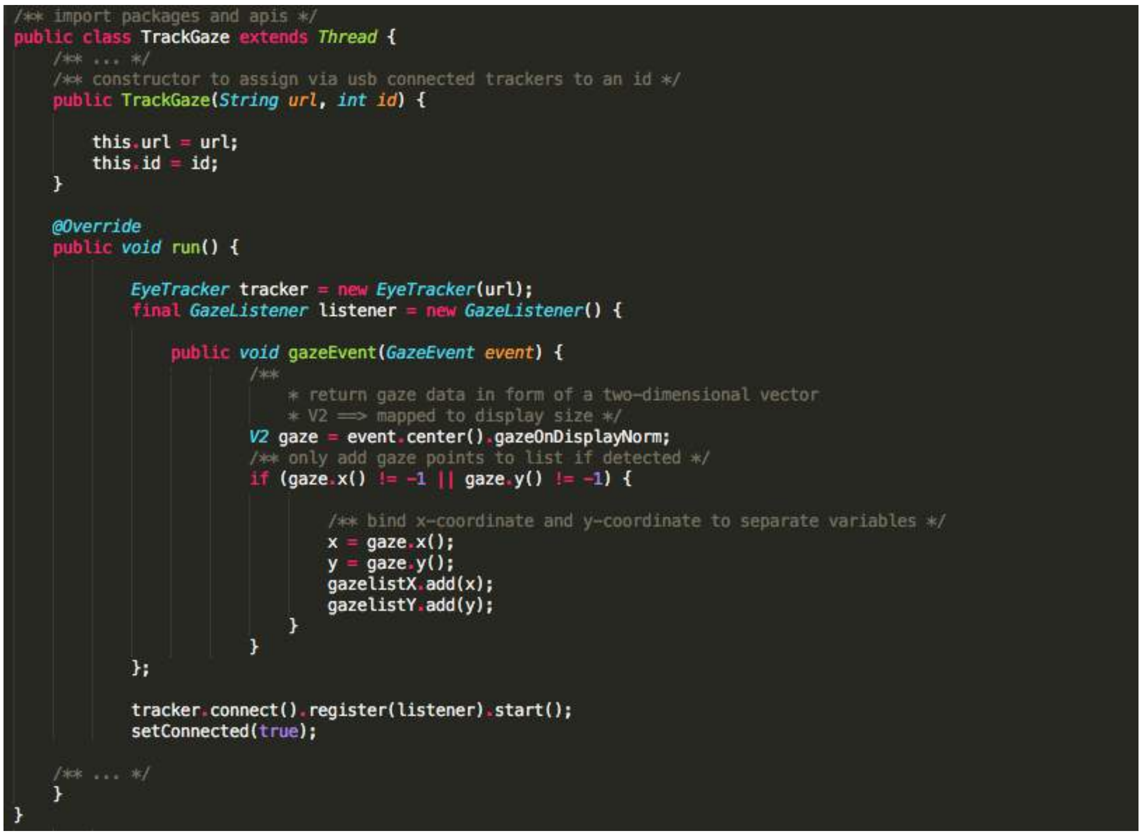
Minim is a useful library for people needing to include sound and audio into their prototypes coded using Processing. It was written by Damien Di Fede with the intention of using the already provided Java Sound, Tritonus and MP3SPI libraries and making them easily accessible for Processing [8]. The idea behind this library was to keep the implementation as simple as possible while still providing maximum flexibility of use. It was included in Processing 2.0, but it was later on decided to provide the library as importable file.
For our purpose the audio library seemed fitting as we only needed an AudioPlayer object within the Processing code snippet that replayed desired audio files at a given time.

5.2 Implementation Process

Designing the interface to display gaze pursuit stimuli and give visual feedback turned out to be a multi-milestone process.
First of all we decided to initially display three moving grey circles on a dark background. The colour of the circles was arbitrarily chosen to keep neutrality and the black background colour helped avoiding longtime strain of the eyes as opposed to staring at a brighter background. The three circles should all move with different trajectories and speeds, so that the correlation calculations can distinguish matches between gaze and object movement more easily.
Since our research problem also involved studying whether cross-interactions would affect speed or error rate of selections, three circles seemed to be the most sensible choice. The circle in the middle would be moving in a circular motion and the two circles to the left and right in a linear trajectory.
Overall, the system should include the following aspects: Firstly, it should be able to track gaze for more than one user. Secondly, it should contain an interface displaying stimuli for users to interact with, as well as the resulting visual feedback. Furthermore, the tracked gaze data should be correlated with the object movement data for each connected user simultaneously to find matches and accordingly display feedback. And finally, the system should be able to display more than one type of visual feedback. The following sections will explain the different milestones overcome to build said system.

5.2.1 First Milestone: Tracking Gaze for Multiple Users

The wrapper code used, to map received Tobii gaze data to Java provides an intuitive setup to connect eye-trackers and receive gaze behaviour information mapped to the attached display size in use. According to the author, several Tobii REX trackers can be connected simultaneously, assumed each tracker is mapped to an EyeTracker()-object. Figure 5.2 shows the simplified process of connecting eye-trackers and running them in Java.

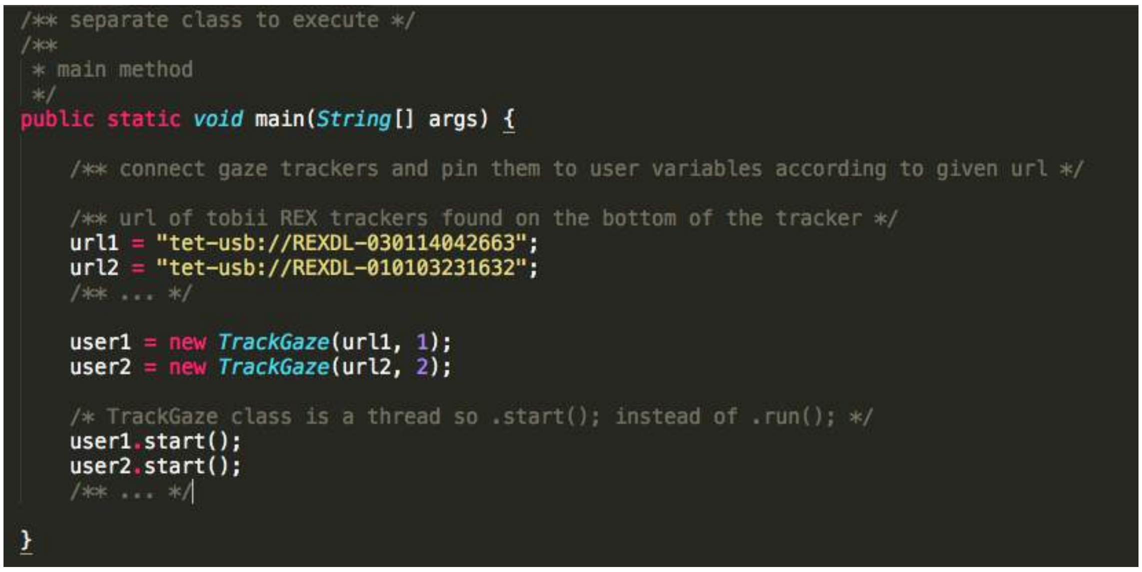


```java
/** import packages and apis */
public class TrackGaze extends Thread {
    /** ... */
    /** constructor to assign via usb connected trackers to an id */
    public TrackGaze(String url, int id) {

        this.url = url;
        this.id = id;
    }

    @Override
    public void run() {

            EyeTracker tracker = new EyeTracker(url);
            final GazeListener listener = new GazeListener() {

                public void gazeEvent(GazeEvent event) {
                        /**
                         * return gaze data in form of a two-dimensional vector
                         * V2 ==> mapped to display size */
                        V2 gaze = event.center().gazeOnDisplayNorm;
                        /** only add gaze points to list if detected */
                        if (gaze.x() != -1 || gaze.y() != -1) {

                                /** bind x-coordinate and y-coordinate to separate variables */
                                x = gaze.x();
                                y = gaze.y();
                                gazelistX.add(x);
                                gazelistY.add(y);
                            }
                        }
            };

            tracker.connect().register(listener).start();
            setConnected(true);

    /** ... */
    }
}
```

Figure 5.2: Connecting eye-trackers and receiving gaze data

The class TrackGaze is representative for each connected eye-tracker, and thus attached to each potential user. Since each user has to process a variety of information passed from the eye-tracker and therefore is required to calculate correlation between object movement and gaze behaviour independently, it seemed profitable to make use of the multi-threading approach. In this idea each TrackGaze()-object is primarily a thread running simultaneously to the main thread every simple program consists of. With this approach a memory bottleneck is avoided as well as blockage of calculating power.
Each eye-tracker contains a URL which can normally be found at the bottom of the Tobii REX trackers. Using this address makes it easier to bind a tracker to a user, by additionally giving the TrackGaze() constructor an id, which is an increasing number based on the amount of users planning to connect to the interface, starting the count at 1.
Having successfully created the TrackGaze()-object, the next step is to fill the run()-method. This method is essential to every thread, as it is constantly recalled and thus acts as the motor behind every functioning thread.
In this run()-method an EyeTracker()-object is created for every user and bound to the given URL. Now we have our eye-tracking device wrapped to the Java code. To receive incoming gaze data, it is necessary to apply an object that acts as a listener, waiting for any appearing gaze

data. In this case that listener is called a GazeListener(), which every eye-tracker bound to an EyeTracking()-object has to add in order to actually obtain information sent by the Tobii REX tracker.
In this listener eye movement coordinates are received in the form of a two-dimensional vector, which contains both the x-coordinate as well as the y-coordinate. The received gaze data in case it is existing, is added to two separate lists for further correlating purposes.
Having added the listener object it is eventually necessary to connect the tracker, register a listener and start the tracker object.
With this setup gaze of multiple users can be potentially tracked, but the whole algorithm only works if these users are actually added to the system. For this matter, the main-method, which is the method that starts the program and is executed at every start of a new application, needs to include the creation and starting of a TrackGaze()-object for every user that should be connected. Figure 5.3 shows an example of how the TrackGaze()-objects are created. Each object receives a different URL as well as a manually increased ID.

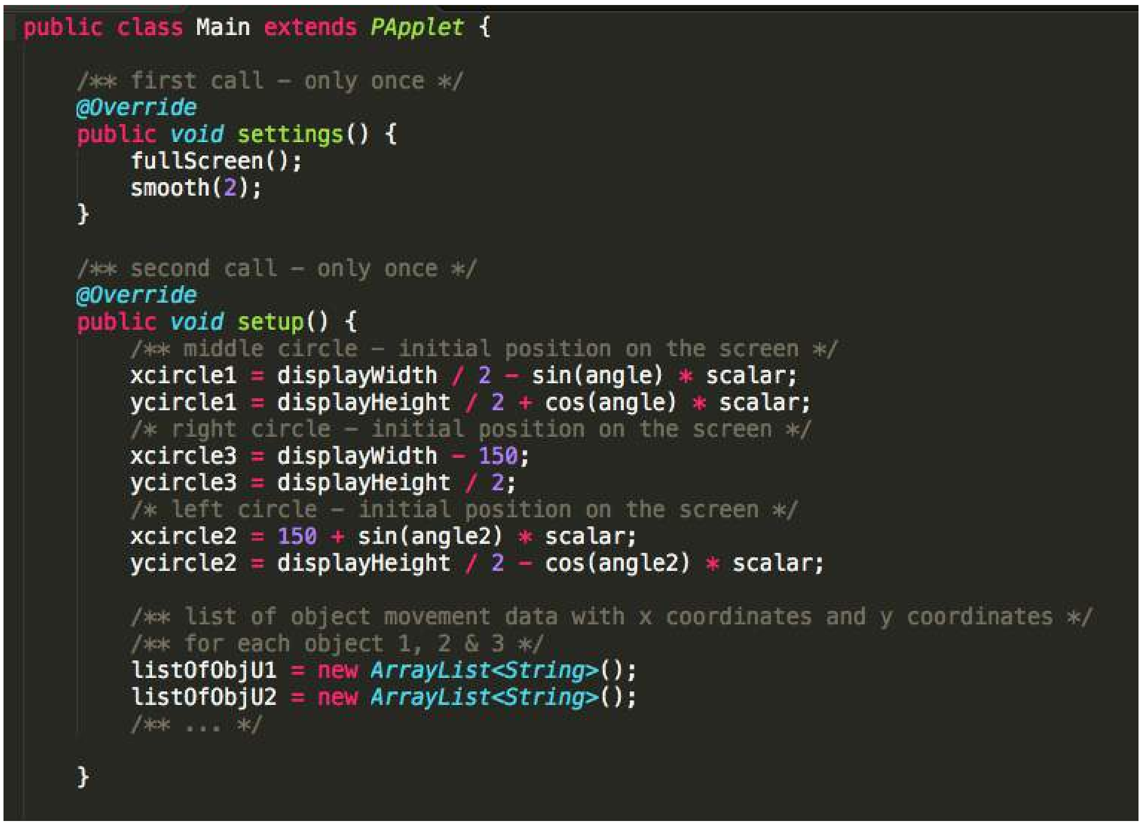


```java
/** separate class to execute */
/**
 * main method
 */
public static void main(String[] args) {

    /** connect gaze trackers and pin them to user variables according to given url */

    /** url of tobii REX trackers found on the bottom of the tracker */
    url1 = "tet-usb://REXDL-030114042663";
    url2 = "tet-usb://REXDL-010103231632";
    /** ... */

    user1 = new TrackGaze(url1, 1);
    user2 = new TrackGaze(url2, 2);

    /* TrackGaze class is a thread so .start(); instead of .run(); */
    user1.start();
    user2.start();
    /** ... */

}
```

Figure 5.3: Connecting eye-trackers and receiving gaze data

5.2.2 Second Milestone: Moving Circles as Stimuli

After successfully integrating the use of eye-trackers into Java and enabling multiple users to connect to the system, the following step was to design the interface containing more than one moving object. In this case we decided to use circles as the moving stimuli. We opted for three moving circles, which were easily implemented using Processing. The coding language can be integrated into simple Java code by importing core libraries and requires the creation of a PApplet - a Processing Applet. Figure 5.4 shows the required methods to initially set up the PApplet.
Every PApplet presumes the existence of the three methods settings(), setup() and draw(). While the draw()-method is called at a default framerate of 60Hz and operates as a thread, the other two methods are normally only called once prior to initialising the applet. In this case we set all the important information in the setup()-method, especially with regards to the x-coordinate and y-coordinate of each moving object as well as the creation of lists containing these coordinates which were used for later correlation purposes.



```
public class Main extends PApplet {

    /** first call - only once */
    @Override
    public void settings() {
        fullScreen();
        smooth(2);
    }

    /** second call - only once */
    @Override
    public void setup() {
        /** middle circle - initial position on the screen */
        xcircle1 = displayWidth / 2 - sin(angle) * scalar;
        ycircle1 = displayHeight / 2 + cos(angle) * scalar;
        /* right circle - initial position on the screen */
        xcircle3 = displayWidth - 150;
        ycircle3 = displayHeight / 2;
        /* left circle - initial position on the screen */
        xcircle2 = 150 + sin(angle2) * scalar;
        ycircle2 = displayHeight / 2 - cos(angle2) * scalar;

        /** list of object movement data with x coordinates and y coordinates */
        /** for each object 1, 2 & 3 */
        listOfObjU1 = new ArrayList<String>();
        listOfObjU2 = new ArrayList<String>();
        /** ... */

    }
```

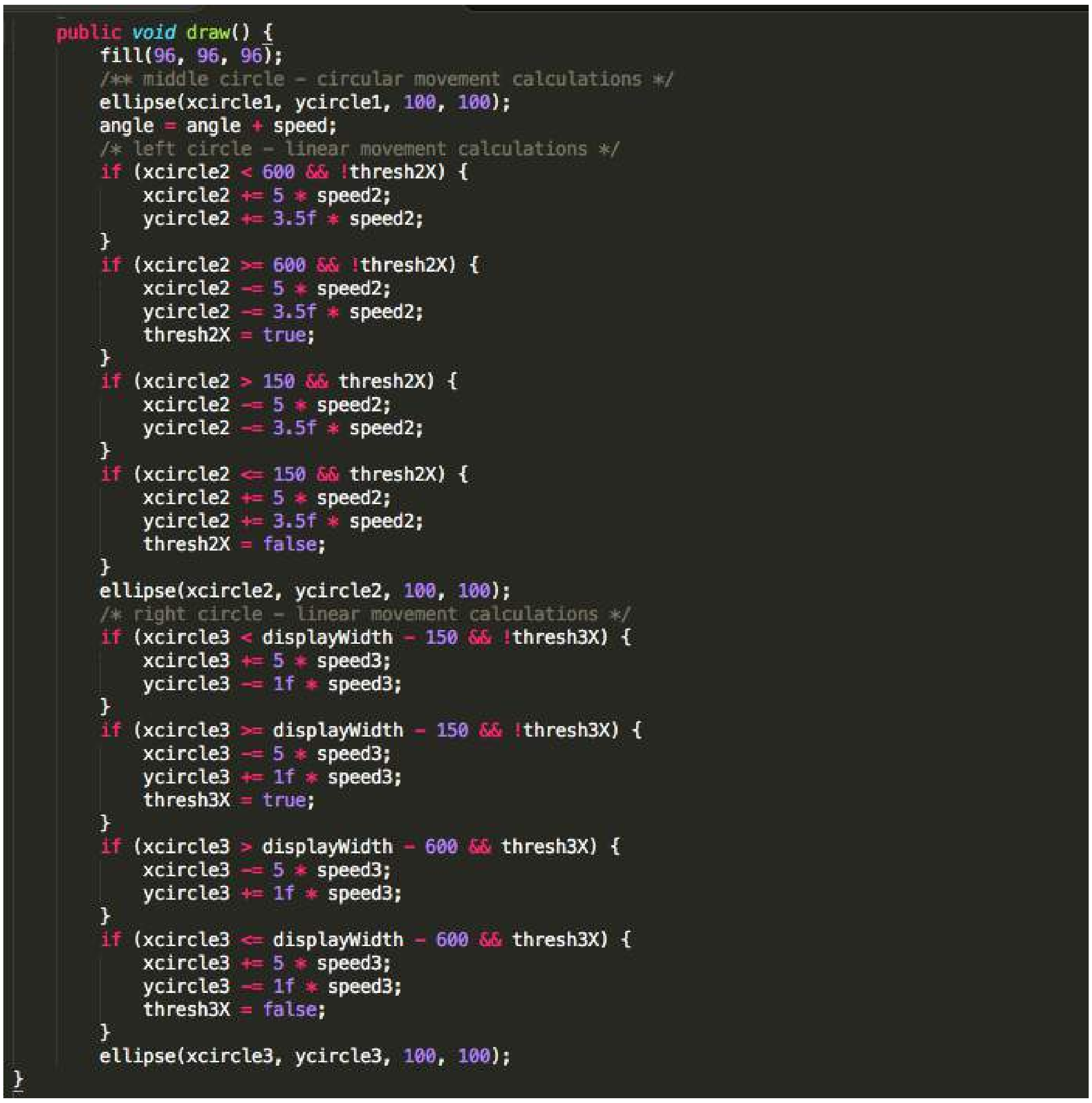
Figure 5.4: Connecting eye-trackers and receiving gaze data

As established previously, the middle circle moves in a circular trajectory. If we want to move the circle in a circular motion we need the two following functions to calculate the x- and y-coordinate respectively (see equations 1 and 2).

$$positionY = positionY + cos(t) * constantVal; \tag{1}$$

$$positionX = positionX - sin(t) * constantVal; \tag{2}$$

In this case the only value changing is t which represents the speed value with which the circle moves, whereas sine and cosine adjust the curve direction the circle takes and thus updates the x- and y-coordinates accordingly. Both equations are interchangeable for the circle movement, as well as subtracting or adding changes whether the circle moves clockwise or counter-clockwise.
As for the linear moving circles to the left and right, the algorithm is conceptualised to let the objects move along a line while respecting certain thresholds vertically and horizontally. The moving space represents a box surrounding the moving circle. Conditions ensure that the circle exclusively moves alongside the given line at a pre-defined speed as seen in 5.5.
The created PApplet is initialised in the main()-method alongside the several users.



```
public void draw() {
    fill(96, 96, 96);
    /** middle circle - circular movement calculations */
    ellipse(xcircle1, ycircle1, 100, 100);
    angle = angle + speed;
    /* left circle - linear movement calculations */
    if (xcircle2 < 600 && !thresh2X) {
        xcircle2 += 5 * speed2;
        ycircle2 += 3.5f * speed2;
    }
    if (xcircle2 >= 600 && !thresh2X) {
        xcircle2 -= 5 * speed2;
        ycircle2 -= 3.5f * speed2;
        thresh2X = true;
    }
    if (xcircle2 > 150 && thresh2X) {
        xcircle2 -= 5 * speed2;
        ycircle2 -= 3.5f * speed2;
    }
    if (xcircle2 <= 150 && thresh2X) {
        xcircle2 += 5 * speed2;
        ycircle2 += 3.5f * speed2;
        thresh2X = false;
    }
    ellipse(xcircle2, ycircle2, 100, 100);
    /* right circle - linear movement calculations */
    if (xcircle3 < displayWidth - 150 && !thresh3X) {
        xcircle3 += 5 * speed3;
        ycircle3 -= 1f * speed3;
    }
    if (xcircle3 >= displayWidth - 150 && !thresh3X) {
        xcircle3 -= 5 * speed3;
        ycircle3 += 1f * speed3;
        thresh3X = true;
    }
    if (xcircle3 > displayWidth - 600 && thresh3X) {
        xcircle3 -= 5 * speed3;
        ycircle3 += 1f * speed3;
    }
    if (xcircle3 <= displayWidth - 600 && thresh3X) {
        xcircle3 += 5 * speed3;
        ycircle3 -= 1f * speed3;
        thresh3X = false;
    }
    ellipse(xcircle3, ycircle3, 100, 100);
}
```

Figure 5.5: Connecting eye-trackers and receiving gaze data

5.2.3 Third Milestone: Correlating Object and Gaze Movement

Correlating the obtained gaze data with simultaneously achieved object movement data turned out to be more difficult than expected.
The gaze data, as previously stated is recorded using Tobii REX eye-trackers and are added into two lists separating x-coordinate and y-coordinate of obtained gaze points. Each user hence has their own gaze lists. Therefore, each user also needs separate lists for object coordinates, as the gaze points are possibly tracked at different times and with a different quantity for each user.
To address distinguished object movement coordinate lists, we created ArrayList<Float>()-objects for each of the three objects and also for each coordinate x and y, resulting in a total of six lists per user in addition to the existing two lists for gaze components. We now have a total of eight lists for each user. To ensure that object coordinates are only added when users receive gaze data, we used AtomicBoolean values called trackerfoundGaze for each user to call and set whenever gaze data is added to the list. This type of boolean is thread-safe and ensures that multiple threads can check and overwrite the value without worrying about thread-safety violation, as this boolean type is updated atomically.

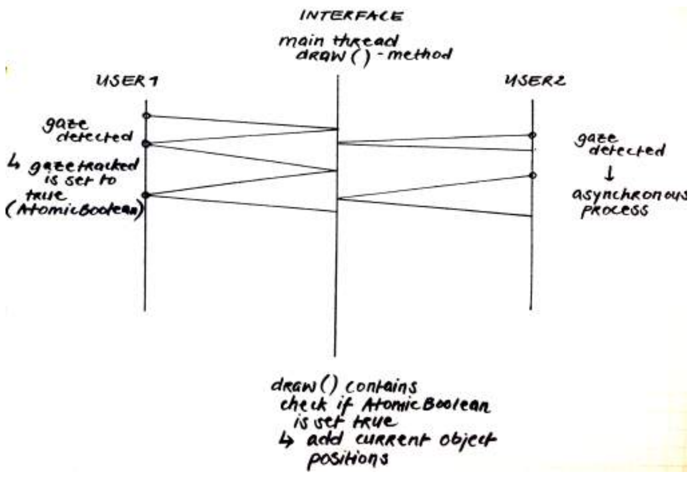
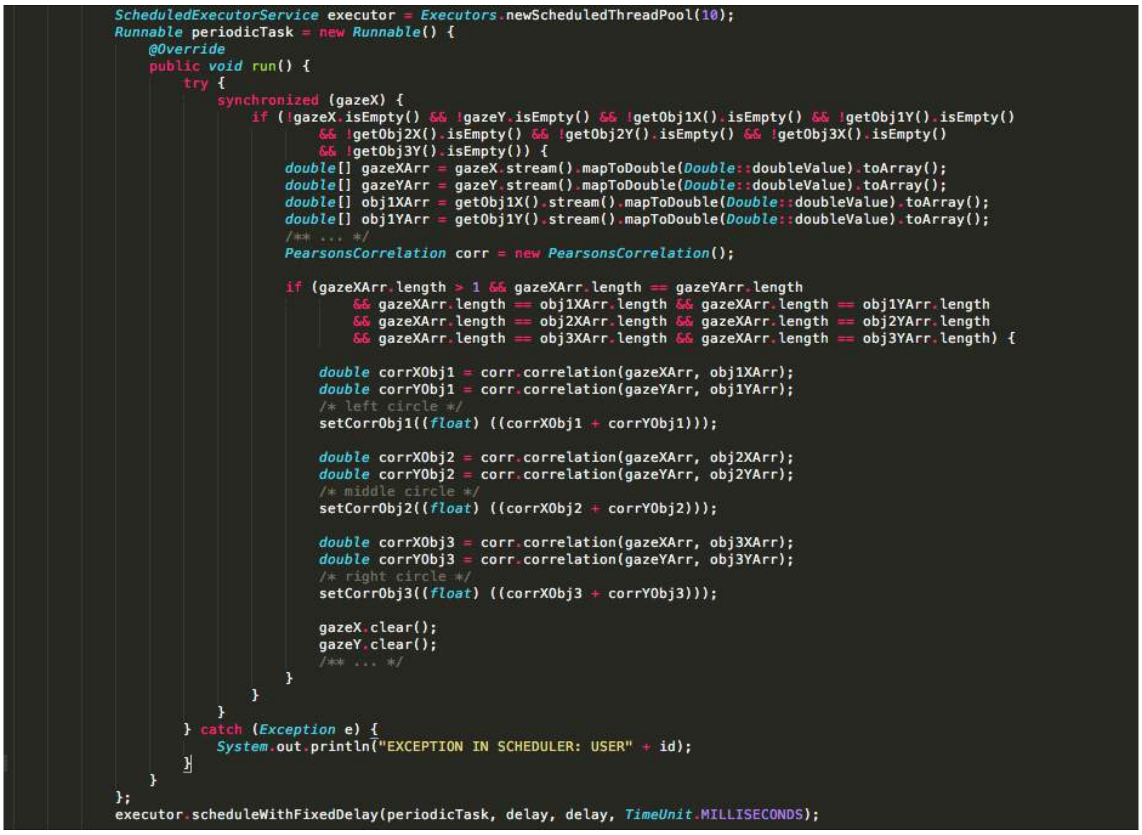

INTERFACE
main thread
DRAW ( ) - method
USER1
USER2
gaze detected
↳ gazetracked is set to true (AtomicBoolean)
gaze detected
↓
asynchronous process
draw() contains check if AtomicBoolean is set true
↳ add current object positions

Figure 5.6: Gaze detection leading to object coordinates being added

Figure 5.6 shows the asynchronous way of obtaining object movement data for multiple users. The AtomicBoolean value for each connected TrackGaze()-object is updated whenever a gaze point is detected and set to true. Since the draw()-method has an if-condition checking with a framerate of 60Hz whether the boolean value is set to true, the current object coordinates are added to their respective lists for the given user. The actual correlating part happens in the user's own TrackGaze()-object. The algorithm is conceptualised to schedule a recurring task every 500ms.
For that matter we used an executor service provided by Java which takes a non-blocked thread out of a pool and executes the calculations specified in its run()-method. Figure 5.7 shows a step-to-step explanation of the correlation algorithm happening every 500ms for each user. First of all one of the eight lists containing data on gazeX, gazeY, object1X, object1Y, etc. is synchronised. In this case we took gazeX. Since setting the AtomicBoolean starts the adding process, which is thread-safe, it blocks other lists from being updated as well, for the duration of calculating the correlation, as the sizes for both samples are required to be the same.
For the purpose of correlating calculations, we made use of the Pearson Correlation Coefficient, a way to show the linear relationship between two datasets of equal size. The correlation value can either take a positive or a negative number between 0 and 1, where the closer the correlation value is to 1, the higher the correlation itself. The value being negative means that gaze and object movement might correlate but in a mirrored way, meaning that e.g. if the object moves downward but the gaze is tracked as moving upwards along the same line the correlation value between those coordinates returns a negative high value.
Before correlations can be calculated the lists need to be converted to arrays, as the correlation function provided by the Google Commons Math library only support arrays as parameter. Following this step a condition checks if any of the lists is empty, as the correlation function further only works if the the samples to be correlated have at least two entries each. If the condition is met that all lists on gaze and object movements have the same size and contain more than one element each, the correlation values are calculated for gaze and the three objects separately. Each object movement and gaze correlation results in two values for the x-coordinate and the y-coordinate. The overall correlation for the respective object would be the sum of those two single correlation values and is set for each user.
The last step is to clear all lists so that they can be re-filled during the next time slot. The executor service is set to check the correlation every 500ms starting after an initial count of 500ms, until the program is closed.



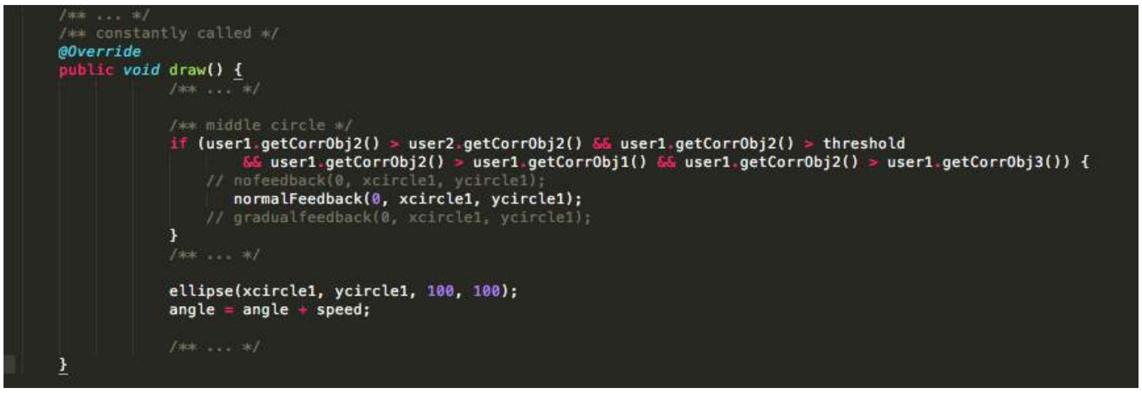
```java
ScheduledExecutorService executor = Executors.newScheduledThreadPool(10);
Runnable periodicTask = new Runnable() {
    @Override
    public void run() {
        try {
            synchronized (gazeX) {
                if (!gazeX.isEmpty() && !gazeY.isEmpty() && !getObj1X().isEmpty() && !getObj1Y().isEmpty()
                        && !getObj2X().isEmpty() && !getObj2Y().isEmpty() && !getObj3X().isEmpty()
                        && !getObj3Y().isEmpty()) {
                    double[] gazeXArr = gazeX.stream().mapToDouble(Double::doubleValue).toArray();
                    double[] gazeYArr = gazeY.stream().mapToDouble(Double::doubleValue).toArray();
                    double[] obj1XArr = getObj1X().stream().mapToDouble(Double::doubleValue).toArray();
                    double[] obj1YArr = getObj1Y().stream().mapToDouble(Double::doubleValue).toArray();
                    /** ... */
                    PearsonsCorrelation corr = new PearsonsCorrelation();

                    if (gazeXArr.length > 1 && gazeXArr.length == gazeYArr.length
                            && gazeXArr.length == obj1XArr.length && gazeXArr.length == obj1YArr.length
                            && gazeXArr.length == obj2XArr.length && gazeXArr.length == obj2YArr.length
                            && gazeXArr.length == obj3XArr.length && gazeXArr.length == obj3YArr.length) {

                        double corrXObj1 = corr.correlation(gazeXArr, obj1XArr);
                        double corrYObj1 = corr.correlation(gazeYArr, obj1YArr);
                        /* left circle */
                        setCorrObj1((float) ((corrXObj1 + corrYObj1)));

                        double corrXObj2 = corr.correlation(gazeXArr, obj2XArr);
                        double corrYObj2 = corr.correlation(gazeYArr, obj2YArr);
                        /* middle circle */
                        setCorrObj2((float) ((corrXObj2 + corrYObj2)));

                        double corrXObj3 = corr.correlation(gazeXArr, obj3XArr);
                        double corrYObj3 = corr.correlation(gazeYArr, obj3YArr);
                        /* right circle */
                        setCorrObj3((float) ((corrXObj3 + corrYObj3)));

                        gazeX.clear();
                        gazeY.clear();
                        /** ... */
                    }
                }
            }
        } catch (Exception e) {
            System.out.println("EXCEPTION IN SCHEDULER: USER" + id);
        }
    }
};
executor.scheduleWithFixedDelay(periodicTask, delay, delay, TimeUnit.MILLISECONDS);
```

Figure 5.7: Gaze detection leading to object coordinates being added

5.2.4 Fourth Milestone: Adding Feedback

After successfully obtaining correlation values for the user, the important step of giving feedback follows. It is essential for our research to give the users multiple types of visual feedback, so we decided to have three types of visual feedback displayed for better comparison. After discussion, we opted to include giving the users the option of no visual feedback as well. However, to keep them informed of rightfully calculated correlations we decided to add different sounds for each user to better distinguish which user made the right selection.
In order to trigger the feedback giving process, we predefined a threshold, the correlation value should reach or surpass. In this case we used 0.9f as the single threshold value to be surpassed for correlations calculated in terms of linear movement. As they are calculated for each coordinate x and y, we take double of that value resulting in 1.8f. Therefore the correlation value for linear movement and gaze movement should surpass or become equal to 1.8f. For circular movements the threshold value was 1.6f.
Furthermore, the condition guarantees that only one object is selected as being rightfully chosen at a time. For that matter, the feedback is only given when firstly, the threshold is met, and secondly, the correlation value for the evaluated object is higher than the ones calculated for the other two objects which is shown in figure 5.8 for the middle object and user1.

In the following the two types of visual feedback are explained further, as no feedback does not change the way users see the display screen, there is no distinctive change in the code.



```java
/** ... */
/** constantly called */
@Override
public void draw() {
        /** ... */

        /** middle circle */
        if (user1.getCorrObj2() > user2.getCorrObj2() && user1.getCorrObj2() > threshold
                && user1.getCorrObj2() > user1.getCorrObj1() && user1.getCorrObj2() > user1.getCorrObj3()) {
            // nofeedback(0, xcircle1, ycircle1);
                normalFeedback(0, xcircle1, ycircle1);
            // gradualfeedback(0, xcircle1, ycircle1);
        }
        /** ... */

        ellipse(xcircle1, ycircle1, 100, 100);
        angle = angle + speed;

        /** ... */
}
```

Figure 5.8: If-clause to check if all conditions are met for feedback

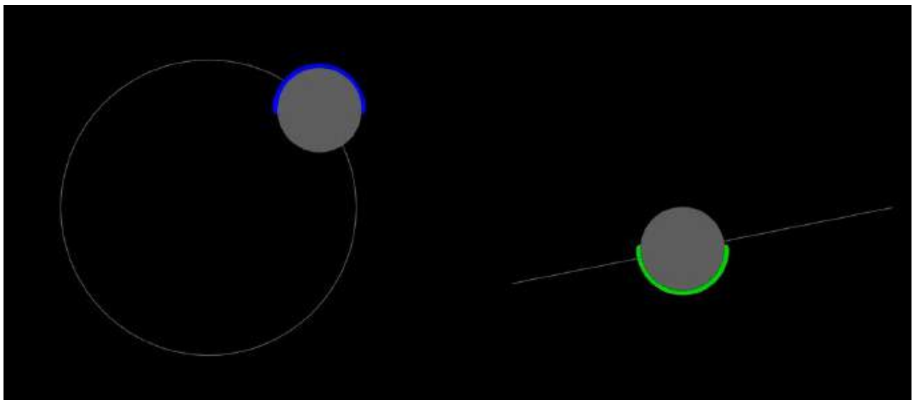
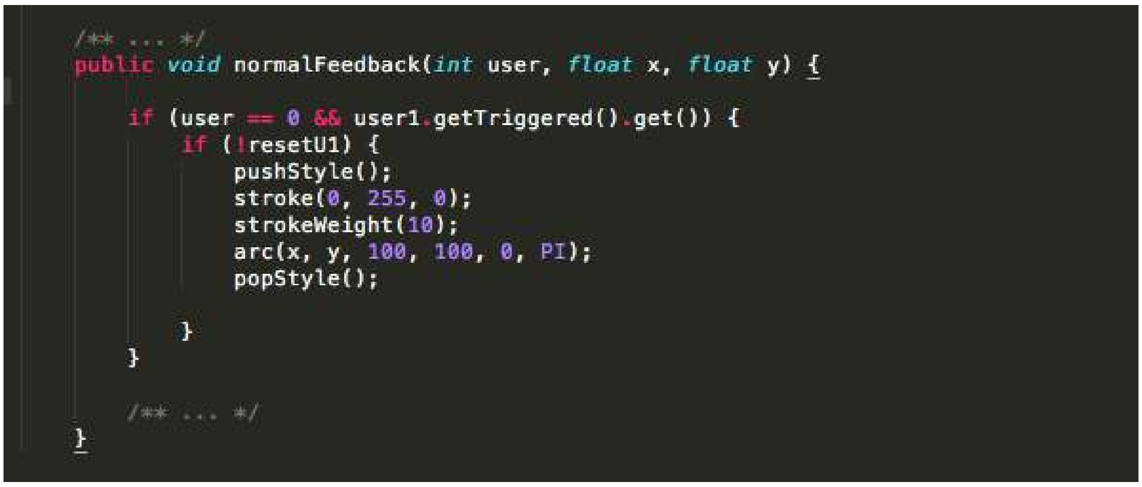


Figure 5.9: Visualisation for normal feedback: arcs of circles wrapping around moving object

Normal Visual Feedback For normal visual feedback the previously mentioned thresholds are sufficient for display of feedback, as the concept behind getting normal feedback marks instant feedback on selection.
As soon as the circle is recognised as selected, an arc of a circle in the respective user's colour appears which is wrapped around the grey moving circle, signalling that the user has selected the given circle, as can be seen in figure 5.9. If more than one user selects the same circle, both appearing arcs are wrapped around the same circle. The arc-method is chosen, as to not give any user the feeling that one circle is *owned* by either one or the other. Instead this type of feedback only hints that one user has been pursuing the moving object.

In order to achieve the arcs of circle seen above, figure 5.10 shows the code snippet to create and colour the arc. Processing provides a function arc(), which receives the current position of the fixated moving circle, as the arc of circle is wrapped around said circle, as well as a radius value that is approximately the circle radius as well. Furthermore, instead of filling out the arc, only the outline is drawn, to achieve the illustration shown above. The calls of pushStyle() and popStyle() are provided by Processing to ensure that the other circles do not have an arc appearing as well, thus only limiting visual feedback to the selected moving object.

The method normalFeedback() is called whenever the correlation value threshold is met for one of the three moving circles by one of the users. The logic to achieve normal feedback for other users happens analogously.



```
/** ... */
public void normalFeedback(int user, float x, float y) {

    if (user == 0 && user1.getTriggered().get()) {
        if (!resetU1) {
            pushStyle();
            stroke(0, 255, 0);
            strokeWeight(10);
            arc(x, y, 100, 100, 0, PI);
            popStyle();

        }
    }

    /** ... */
}
```

Figure 5.10: Processing code to achieve normal visual feedback

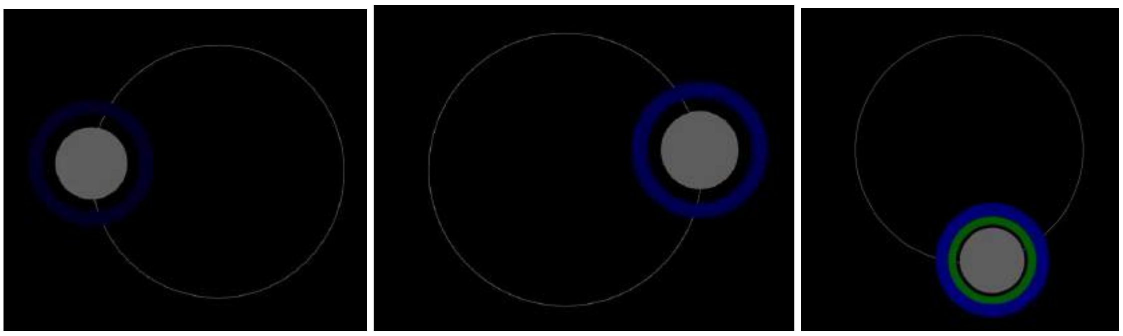
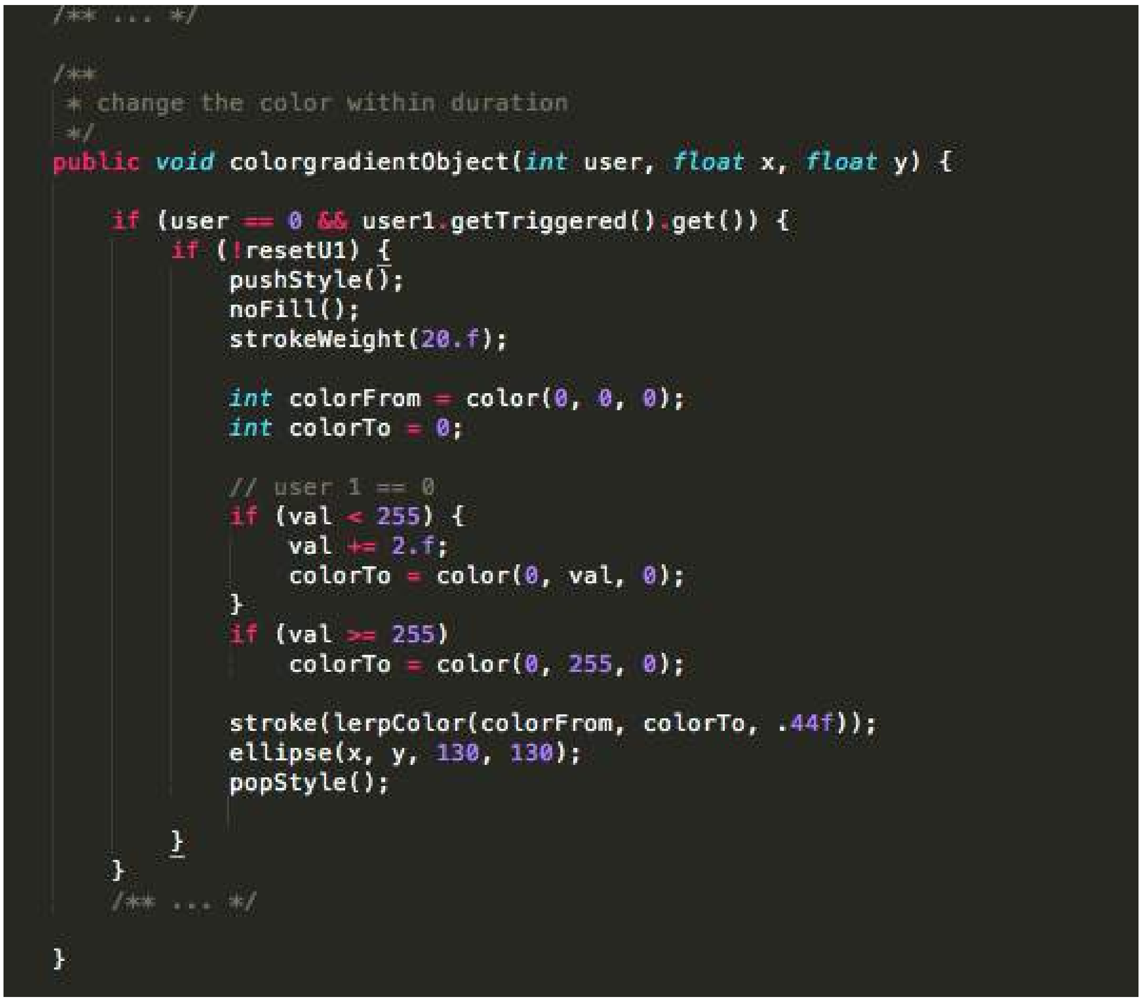


Figure 5.11: Visualisation for gradual feedback: circle around moving object gradually filling with colour

Gradual Visual Feedback As for gradual visual feedback, the recognition process is different. Initially the user's calculated correlation values have to hit the threshold mark, which triggers the start of gradual feedback. Over a course of 180 frames, which at a framerate of 60Hz result in 3 seconds, the gradual feedback is given.
According to figure 5.11 a hollowed circle appears around the currently triggered object and gradually over time fills out the opacity of the colour assigned to the specific user. When the colour is fully opaque, the circle was successfully selected by the user.

Behind the visualisation the system waits for a second trigger after the initial threshold of 1.6f and 1.8f is met. If in the close future a second threshold of 1.2f for the same object triggered is reached with another calculated correlation value unlike the initial trigger correlation, it is counted as the start for gradual feedback. Basically a count variable is incremented and if this count is higher than 2 then the visualisation part starts. Figure 5.12 illustrates the gradual gain of opacity. The code is surrounded by a call of pushStyle() and pushStyle() to ensure that only the moving object that is supposedly currently selected and therefore triggered, gets the visual change. The idea behind this gradual change is to gradient a colour from black and thus opaque to the background to the assigned colour for the user, in this case green. The only value increased is a count that acts as a colour value. If this value exceeds 255, the maximal value RGB colours can reach per component, then the value is no longer updated but continues to be displayed in said colour, thus signalling that the circle is selected.
In addition to the visual feedback the audio feedback, as mentioned earlier, is played back at the



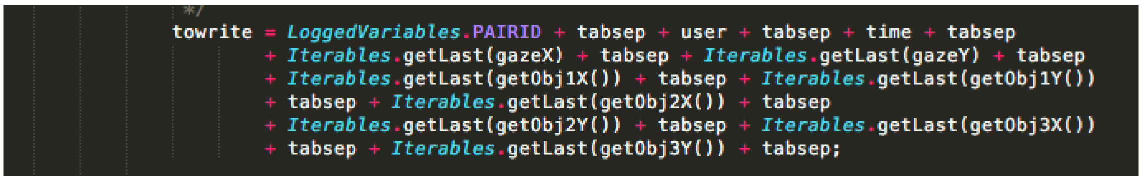
```
/** ... */

/**
 * change the color within duration
 */
public void colorgradientObject(int user, float x, float y) {

    if (user == 0 && user1.getTriggered().get()) {
        if (!resetU1) {
            pushStyle();
            noFill();
            strokeWeight(20.f);

            int colorFrom = color(0, 0, 0);
            int colorTo = 0;

            // user 1 == 0
            if (val < 255) {
                val += 2.f;
                colorTo = color(0, val, 0);
            }
            if (val >= 255)
                colorTo = color(0, 255, 0);

            stroke(lerpColor(colorFrom, colorTo, .44f));
            ellipse(x, y, 130, 130);
            popStyle();

        }
    }
    /** ... */

}
```

Figure 5.12: Processing code to achieve gradual visual feedback

time of selection and acts as a second guarantee that the user is informed about their selection.

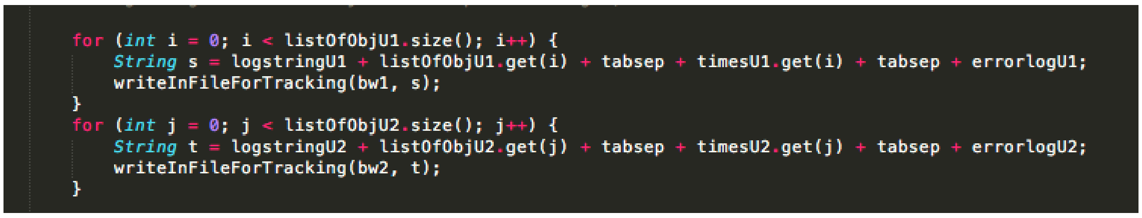
5.2.5 Fifth Milestone: Prepare Logging Process

Having the interface ready to display visual feedback if needed and triggered, it was essential for our study to include logging opportunities as part of our user study design. Logging is divided into two parts, one being the gaze behaviour in general was logged for each user, and the other part was logging the user's selections.
As for gaze logging, the x- and y-coordinate of each gaze point was logged as well as the time in ms at which the movement point was tracked by the Tobii REX. Furthermore the user ID was logged as well as the object x and y positions at the time as can be seen in figure 5.13. Each value was separated using the tabulator, this was decided in order to facilitate the import of the .txt file into Excel for evaluation purposes.
On the other hand logging object matches was more complex and divided into multiple parts. First of all, prior to starting the task user ID, group ID and start time are saved in respective variables. Furthermore the expected object is logged for each user and the type of feedback and the condition of selection was logged. These values were later determined in task files. During the interaction process we would write into the log each time a selection was made, regardless of whether said selection was a match to the expected object or not and we would save that object in a list with the recognition time, i.e. the time it was marked as selected or in case of gradual feedback, the time it



```
            */
            towrite = LoggedVariables.PAIRID + tabsep + user + tabsep + time + tabsep
                    + Iterables.getLast(gazeX) + tabsep + Iterables.getLast(gazeY) + tabsep
                    + Iterables.getLast(getObj1X()) + tabsep + Iterables.getLast(getObj1Y())
                    + tabsep + Iterables.getLast(getObj2X()) + tabsep
                    + Iterables.getLast(getObj2Y()) + tabsep + Iterables.getLast(getObj3X())
                    + tabsep + Iterables.getLast(getObj3Y()) + tabsep;
```

Figure 5.13: Log string for gaze and object position logging per user



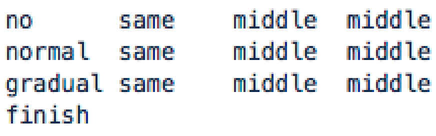
```
    for (int i = 0; i < listOfObjU1.size(); i++) {
        String s = logstringU1 + listOfObjU1.get(i) + tabsep + timesU1.get(i) + tabsep + errorlogU1;
        writeInFileForTracking(bw1, s);
    }
    for (int j = 0; j < listOfObjU2.size(); j++) {
        String t = logstringU2 + listOfObjU2.get(j) + tabsep + timesU2.get(j) + tabsep + errorlogU2;
        writeInFileForTracking(bw2, t);
    }
```

Figure 5.14: Final log string for selection tasks per user if task is over

was recognised as being in train of being selected.
If the match was made we saved the end time as the final time to later calculated the time needed to select the circle given in the task. Additionally, we added the error rate to the log. Errors were counted every time the user hits the respectively assigned key to cancel or reset their selection which they were informed about. We would include this value into the log es well.
Unfortunately we did not count the number of times the user was distracted by their partner, which we had to manually dissect after the study ended, but in theory the number of distractions would be incremented, whenever the user selected an object that their partner had assigned to as their expected object at that time, which would result in a comparison of the partner's expected and the user's currently looked at moving circle.
Figure 5.14 illustrates the final log string and the resulting output. All logging was done using the BufferedWriter() and FileWriter()-objects provided by Java which were setup prior to interaction and closed after logging mechanisms were shut down.

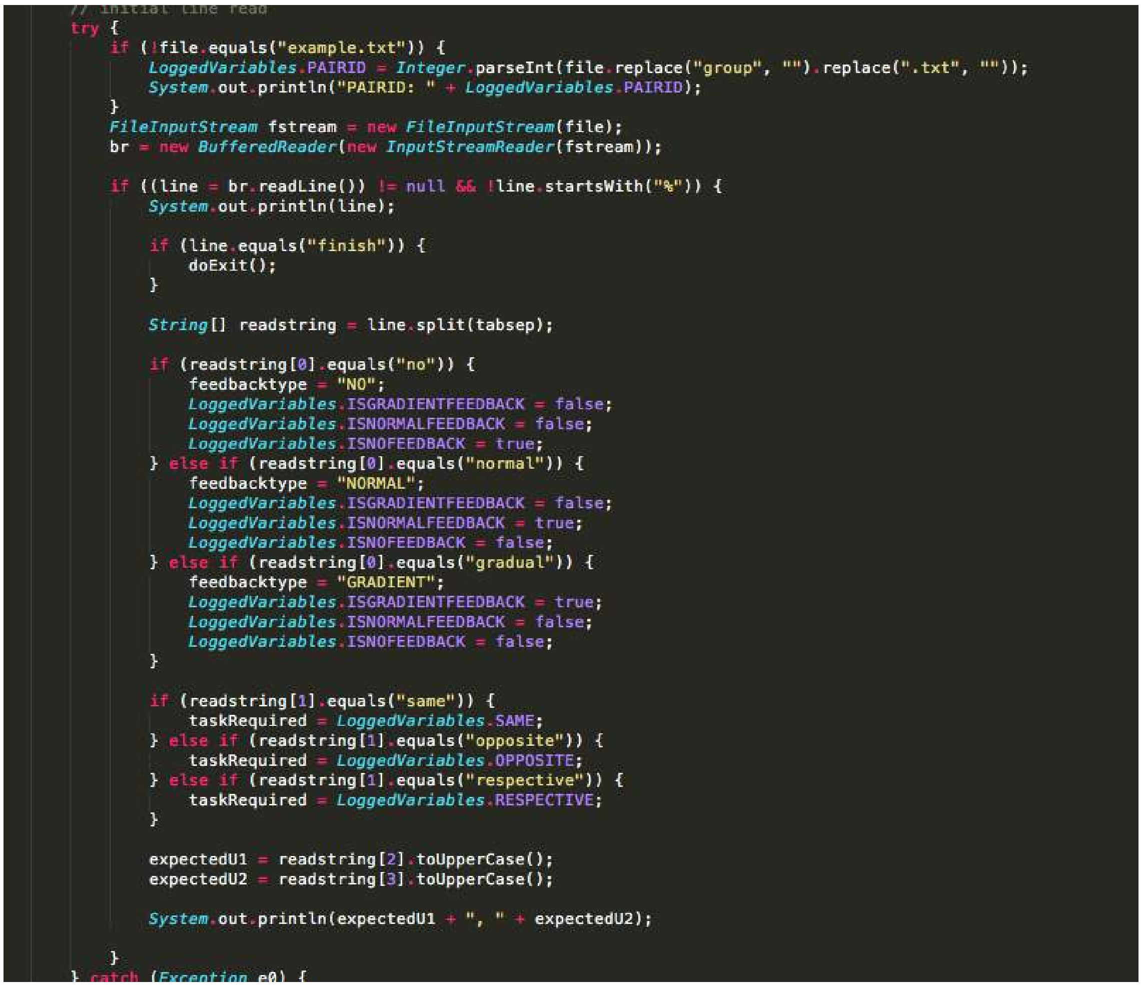
5.2.6 Sixth Milestone: Loading Study Tasks

The final step was to include how participants were told which object they were supposed to look at. For this purpose we typed the selection tasks into .txt files for easier separation and not having to tediously include them into code scenarios. Each .txt file contains the type of feedback the participants are currently expecting, which object user1 is expected to select and which object user2 is expected to pursue, as can be seen in figure 5.15. The figure shows a snippet of a task file. The file is read line by line using BufferedReader() and FileReader(). The next line is only read if the task in the currently read line is completed. The line elements are divided into an array of strings and the values are parsed accordingly, i.e. if the task is delivering gradual feedback then a final constant variable ISGRADIENTFEEDBACK is set to true, whereas the other constant booleans ISNORMALFEEDBACK and ISNOFEEDBACK are kept false. Also there are variables for each user called saving the object they are expected to look at. The task is printed above the



```
no      same    middle  middle
normal  same    middle  middle
gradual same    middle  middle
finish
```

Figure 5.15: List of tasks per group of two participants



```java
// initial line read
try {
    if (!file.equals("example.txt")) {
        LoggedVariables.PAIRID = Integer.parseInt(file.replace("group", "").replace(".txt", ""));
        System.out.println("PAIRID: " + LoggedVariables.PAIRID);
    }
    FileInputStream fstream = new FileInputStream(file);
    br = new BufferedReader(new InputStreamReader(fstream));

    if ((line = br.readLine()) != null && !line.startsWith("%")) {
        System.out.println(line);

        if (line.equals("finish")) {
            doExit();
        }

        String[] readstring = line.split(tabsep);

        if (readstring[0].equals("no")) {
            feedbacktype = "NO";
            LoggedVariables.ISGRADIENTFEEDBACK = false;
            LoggedVariables.ISNORMALFEEDBACK = false;
            LoggedVariables.ISNOFEEDBACK = true;
        } else if (readstring[0].equals("normal")) {
            feedbacktype = "NORMAL";
            LoggedVariables.ISGRADIENTFEEDBACK = false;
            LoggedVariables.ISNORMALFEEDBACK = true;
            LoggedVariables.ISNOFEEDBACK = false;
        } else if (readstring[0].equals("gradual")) {
            feedbacktype = "GRADIENT";
            LoggedVariables.ISGRADIENTFEEDBACK = true;
            LoggedVariables.ISNORMALFEEDBACK = false;
            LoggedVariables.ISNOFEEDBACK = false;
        }

        if (readstring[1].equals("same")) {
            taskRequired = LoggedVariables.SAME;
        } else if (readstring[1].equals("opposite")) {
            taskRequired = LoggedVariables.OPPOSITE;
        } else if (readstring[1].equals("respective")) {
            taskRequired = LoggedVariables.RESPECTIVE;
        }

        expectedU1 = readstring[2].toUpperCase();
        expectedU2 = readstring[3].toUpperCase();

        System.out.println(expectedU1 + ", " + expectedU2);

    }
} catch (Exception e0) {
```

Figure 5.16: Load tasks lists into program

moving objects and the system stops movement until the ENTER key is hit to resume the logging process.
Prior to resuming movement and logging the lists are cleared of remaining gaze data. Figure 5.16 shows how the task lists for each group is loaded into the program. Finish signals that the task list is completed and shuts down the program after writing the contents into the log file.

6 User Study

With the graphical interface displaying the three feedback possibilities integrated in an application and ready for use, the following step was to put the system to a test. On these grounds we designed a user study to help understand and record how the several visual feedback methods were perceived by users, as well as to collect qualitative and quantitative data on which feedback type provided a more comfortable interaction. The subsequent chapter sums up the entirety of the user study settings, including the final design, which technical hardware was necessary and how the resulting repeated measures study conducted in a lab environment was accomplished in totality.

6.1 Study Design

Prior to conducting the study it was important to firstly consider which data should be logged for further evaluation purposes, as well as to secondly brainstorm how we should approach realising a user study in general. Additionally what type of study would be fitting in our case also had to be decided.
To recall the contents of the preceding chapter we prepared the following aspects: the interface showing three moving circles, two of which were moving in a linear trajectory and the third one in a circular manner. Furthermore, we had implemented the algorithm to check for correlations between gaze data and object movement, as well as normal and gradual visual feedback, with the intention of only integrating those two types into our study.
The purpose of our user study would primarily lie in testing how visual feedback can be given to and received by multiple users rather than focusing on having them interact on a public display. Bearing in mind space and hardware availability, we further limited the *multiple* user aspect to two simultaneously interacting users at a time and place. The study would take on the form of a repeated measures procedure, i.e. users would be asked to perform a certain task several times with slight alterations. Furthermore the study was to be conducted on a laptop as opposed to a larger display as it was easier to adjust the eye-tracking devices to fit the field of vision of a sitting person rather than having to change the tracker's angle according to height of potential users.
A group discussion encouraged us to add a third visual feedback condition to the picture - no visual feedback. This feedback type was proposed to better test the user's opinion on the existing feedback methods as well as to act as a control condition during the study procedure. Also, we decided on adding different sounds to help each user recognise the algorithm's rightfully made matches between expected and pursued object.

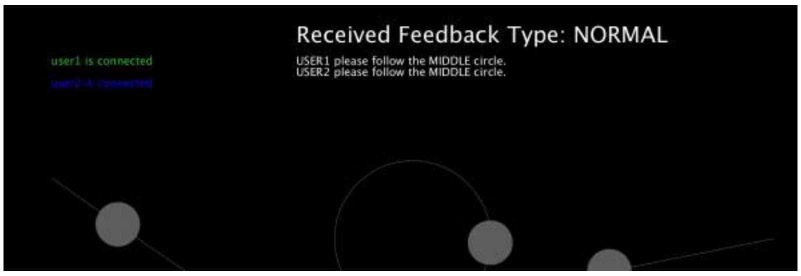
IV	(a)	(b)	(c)
(1) feedback type	NO	NORMAL	GRADUAL
(2) selection type	SAME (both users look at the same object)	OPPOSITE (each user looks at an object closer to their partner)	RESPECTIVE (each user looks at an object closer to them)

DV	(1)	(2)	(3)
	SPEED (duration until selection match was made)	ERRORS (numbers of times the system recognised the wrongly selected object)	DISTRACTIONS (numbers of times one user looked at object their partner had to select)

Table 6.1: Table summarising of Independent Variables (IV) and Dependent Variables (DV)

Following this decision we defined two independent conditions at three types each, which are also

illustrated in table 6.1. On the one hand, independent variable number one was the feedback type, which again consists of the three different feedback possibilities available for the participants: no, normal and gradual feedback. On the other hand, the selection conditions same, opposite and respective object formed the second independent variable.
Thus, every group of two users had to perform three types of selections during each stage: they should both pursue the same object, an object that was closer to each of them and respectively an object that was closer to their partner. For each selection type we were opting for three selection tasks. This would result in 27 selections in total. We also aimed for 24 participants, which would result in 12 groups with two simultaneously interacting people. Having confirmed this number of participants would eventuate in an entirety of 648 selections made.
The actual study would be proceeding in the following way: participants would be required to arrive in pairs, for each interaction session needed two users at a time. For IV (1), the first four groups would start with condition (a), receiving no visual feedback, the following four groups with condition (b), normal feedback and the rest with condition (c), obtaining gradual feedback. As for IV (2) the three types of selection condition tasks with three selections made per task were counterbalanced using a 3x3-Latin-Square for the 12 groups. For easier referral, each user would be assigned a colour and a sound, the colour in this case being green for user1 and blue for user2, as well as the sounds for user1 being a triangle and a gong for user2. The different visual feedback types would be displayed using these colours. Additionally, the task each user had to perform would be written on top of the screen indicating which user was to pursue which moving object for the given task, as shown in figure 6.1. If a match was recognised the particular user was to be disconnected from further logging and feedback was to be given accordingly. Each user would have a specifically designated key to reset their selection at any point. The experimenter would be able to manually shift between selection tasks, in case any problem occurred with either logging or giving feedback. Between each task the interface movement would pause and be reactivated using another specific key.


Received Feedback Type: NORMAL

user1 is connected

user2 is connected

USER1 please follow the MIDDLE circle.
USER2 please follow the MIDDLE circle.

Figure 6.1: Tasks illustrated on top of the interface as well as which user expects which colour and which objects users have to look at

Based on these independent variables and tasks the study was aimed to record three types of dependent variables, which are listed in table 6.1. In order to compare the different visual feedback types and selection conditions efficiently in a quantitative manner, the system should log the speed with which the expected object was selected under the given condition, as well as the number of times the algorithm failed to recognise the rightfully selected object, i.e. the selection error rate, and the number of times each user was distracted by their partner's choices, i.e. the times they accidentally selected the object their partner was expected to follow.

In addition to the hands-on part, a questionnaire was designed to ask participants about their general emotions and thought processes during the study. This was necessary to see whether participants overall would prefer one type of feedback or selection condition over the remaining others. The questionnaire would consist of three main partitions, the first part inquiring information about the participant in general, covering questions ranging from age, gender and eyesight to whether the particular participant had prior experience with gaze interaction or even pursuit interaction. The second part of the questionnaire on the other hand, asked the participants directly on their opinions regarding the feedback technique they were just shown.

6.2 Technical Setup

The previously described user study was conducted using a laptop of the Lenovo T450p series, consisting of a 15.6 inch screen, which for the purpose of the repeated measures study was set to a resolution of 1080p full HD, i.e. 1920x1080 pixels. The specifications further entailed a Windows 8.1 Enterprise 64-bit OS, with an i7-4800MQ CPU (2.70GHz) and an 8GB memory usage availability.
Connected to said laptop were two Tobii REX eye-trackers, each attached to a tabletop mini tripod, with a gaze data sample rate of approximately 30Hz, which were placed laterally to the prior mentioned device.

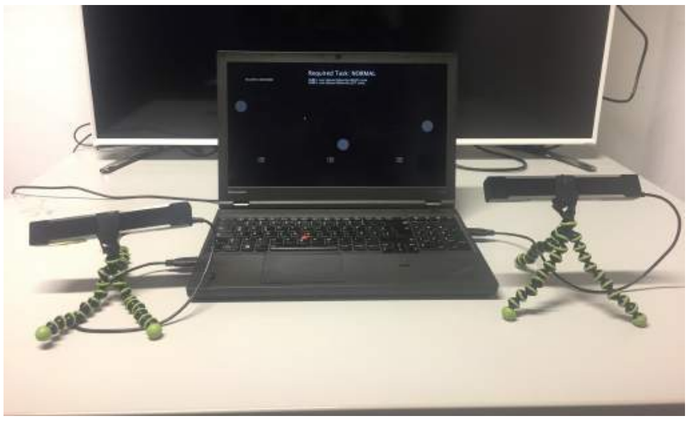


Figure 6.2: Technical setup for user study: Laptop with two attached Tobii REX

Figure 6.2 shows the final setting for the lab study. The position of the eye-trackers was purposefully decided in order to not obstruct the user's view on the display and thus the interaction interface, whereas the laptop was chosen for the study, mainly because of its OS supporting the Tobii SDK as well as the display size being big enough for more than one user to interact comfortably.
Following study intent, gaze data was logged, while taking into account temporal and spatial information. Additionally, the application was designed to save each correctly recognised object-gaze-movement match with selection speed and overall error rate.
In order for participants to fill out the accompanying questionnaire without disrupting the technical setup, two additional laptops were provided: a Microsoft Surface Pro 4 and an Apple MacBook Pro. Essentially these laptops were equipped with a browser being able to load Google Forms.
Figure 6.3 shows in general which keys were assigned for what purpose. The keys 1, 2 and 3 were used to connect users in case the automatic process did not work as desired (2 and 3), as well as to skip forward to the next selection task (key 1), in case the system did not give any indications that a selection match was made successfully, i.e. no feedback was given. For user1, the left participant, key W was important to cancel their selection and reset it, which was analogously for user2, the right user, if they pressed the key P. The ENTER key served as start to log the respective task and start the animated interface again between the tasks given. This was implemented to facilitate logging and load the next task.

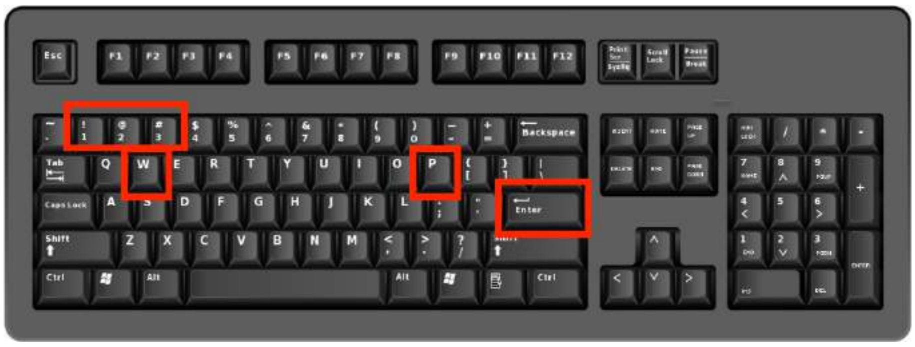


Figure 6.3: Assigned keys for logging purpose

6.3 Study Procedure

As the study specifications required two users per session, a doodle poll was helpful in finalising appointments. On the other hand, experience or prior knowledge about working with eye-trackers, or on gaze pursuit was no obligation.
Upon arrival both participants were briefed on the content and purpose of the repeated measures study they were about to participate in. The explanation contained information on the intent of conveying different types of visual feedback to users applying gaze pursuit interaction and hence getting their opinion on them, as well as stressing that their private data is handled cautiously and that the system is being tested rather than the user themselves. For the majority of the participants a rough illustration on how the eye-trackers work, as well as on what gaze pursuit consists of, was given additionally.
After each signing the declaration of consent, both users were asked to sit down in front of the setup. Furthermore, they were calibrated separately using the standard Tobii EyeX tool. A new gaze profile was created respectively for each of the two participants: user1 and user2. For continuity reasons the participant sitting in front of the left eye-tracker was assigned to the profile user1 and the other one to user2 as illustrated in figure 6.4.

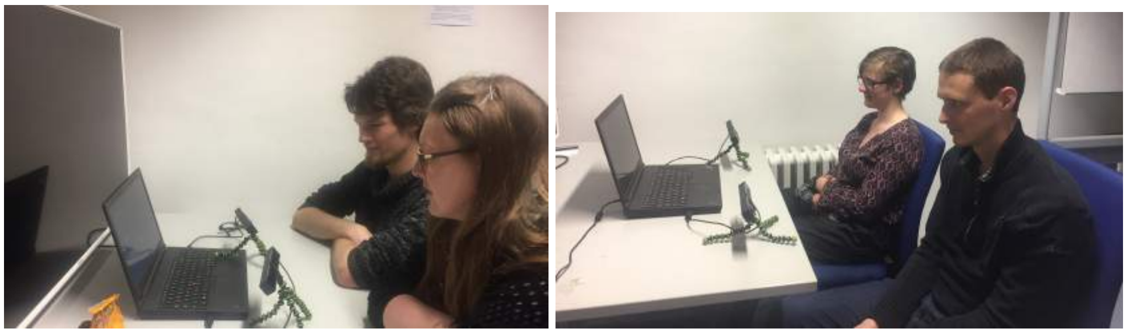


Figure 6.4: First part of study process: Participants' interaction with the interface

Prior to the actual study task, participants were told to expect three types of visual feedback: NO, NORMAL and GRADUAL visual feedback. Following this announcement, participants were given an example of the study interface, for them to see how the particular feedback types appeared on the screen. It was pointed out, that the instructions containing which object their gaze should follow, was written above the three displayed moving circles, as well as which colour and user was assigned to whom.
Furthermore, they were informed that each participant had their own sound indicating the right selection being recognised by the system, as well as which key to press, should they consider

Figure 6.5: During the user study: Participants have to possibility to reset the selection process

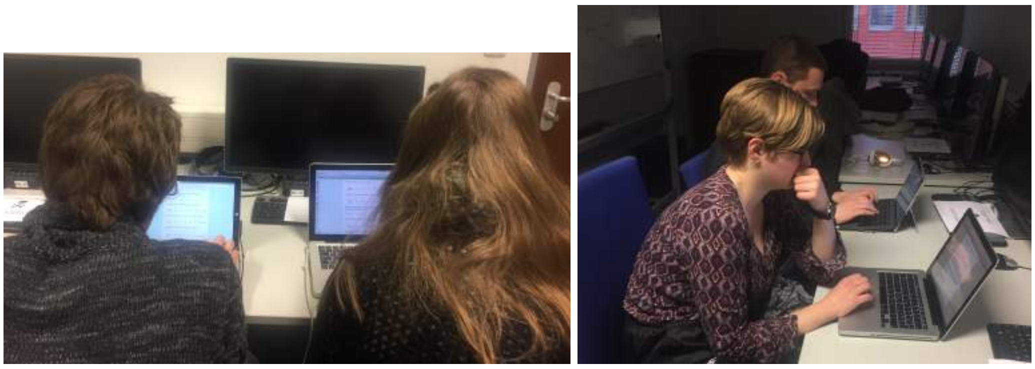
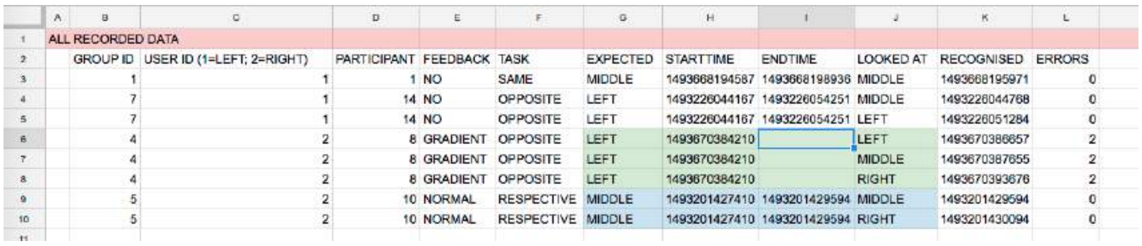


Figure 6.6: Second part of study process: Users answering questionnaires on separate laptops to not disturb the setting and avoid time shortage

resetting the system's calculated object-gaze-movement match as seen in figure 6.5. After each selection task group, which was divided into the three feedback types, both participants were required to fill out the respective questionnaire part to the feedback type they just received. After completing the tasks given to them, they were also asked for a quick ranking of the feedback types and selection conditions they had to fulfill, as can be seen in figure 6.6. We encouraged them to discuss while interacting, whether or not they found something they liked or disliked about the current feedback type or selection condition and tried to observe each user on any subconscious behaviours they had, which might be interesting for evaluation purposes. After completing the study, we often asked them one question about the procedure and whether they could imagine this being helpful in a real-life scenario, as well as what real-life scenarios they could think of, which often led to both participants keeping on the discussion, which we tried to keep a hold of using keywords and catchwords.

7 Evaluation And Results

Following the conduct of the previously explained user study, the process of evaluation and interpretation of the results will be illustrated during the course of the subsequent chapter. For better understanding, whenever the object recognised as looked at by the system is equal to the one expected to be looked at given the task at hand, we will speak of a *match*.
As specified prior to the execution of the study, the system simultaneously logged each participant's gaze behaviour with occurrence time in ms, mapped screen coordinates received by the eye-tracker, and correlation values between gaze and object movement, calculated every 500ms. Another log file also stored the objects after recognition as looked at, with their tracked time. This



	A	B	C	D	E	F	G	H	I	J	K	L
1	ALL RECORDED DATA											
2		GROUP ID	USER ID (1=LEFT; 2=RIGHT)	PARTICIPANT	FEEDBACK	TASK	EXPECTED	STARTTIME	ENDTIME	LOOKED AT	RECOGNISED	ERRORS
3		1	1	1	NO	SAME	MIDDLE	1493668194587	1493668198936	MIDDLE	1493668195971	0
4		7	1	14	NO	OPPOSITE	LEFT	1493226044167	1493226054251	MIDDLE	1493226044768	0
5		7	1	14	NO	OPPOSITE	LEFT	1493226044167	1493226054251	LEFT	1493226051284	0
6		4	2	8	GRADIENT	OPPOSITE	LEFT	1493670384210		LEFT	1493670386657	2
7		4	2	8	GRADIENT	OPPOSITE	LEFT	1493670384210		MIDDLE	1493670387655	2
8		4	2	8	GRADIENT	OPPOSITE	LEFT	1493670384210		RIGHT	1493670393676	2
9		5	2	10	NORMAL	RESPECTIVE	MIDDLE	1493201427410	1493201429594	MIDDLE	1493201429594	0
10		5	2	10	NORMAL	RESPECTIVE	MIDDLE	1493201427410	1493201429594	RIGHT	1493201430094	0
11												

Figure 7.1: Log file entries with colour-marked problematic areas

log further contained the group ID, the user ID, the participant ID, what type of feedback was expected, which type of task was performed, the task starting time, ending time, the time when a certain object was recognised, as well as the count of errors made by the system.
For better comprehensive view, figure 7.1 shows how the entries for the specific log look like. There are two main concern characteristics found in the unclean dataset which are highlighted by colours in figure 7.1: The first problem is missing end time values. Affected rows are marked with the colour *green*. This particular appearance can be noticed if the system fails to ascertain that feedback was supposed to be given following a successfully made match between expected and looked at object.
We can notice a match between the actual object looked at and the expected one in the first row. This observation is also made in the second main concern, where the affected rows are highlighted in the colour *blue*. Supposedly this phenomena happens, when the prior task also requires the specific object to be matched and said task was completed successfully beforehand. In this case, although all gaze and correlation data is reset, the system calculates a match right away, but still continues to log further objects being looked at.

The resulting dataset originally contained 1220 entries, but since there are concern characteristics found throughout the entirety of the log, the first step before commencing the evaluation part, was a clean-up and division of the data into the respective subsets, according to the independent variables: visual feedback type (*no*, *normal* and *gradual*) and selection condition (*same*, *opposite* and *respective*).
We assume that the match between the expected object and the one currently looked at marks the last entry row for a certain task, therefore all the recognised objects for the given task logged after the match, are to be neglected and removed. Furthermore, if there is no end time given, but a match is found, we will presume that this match is shown correctly and calculate an end time by adding 3000ms to the recognition time. This particular time value was preset as the time given for any type of feedback to occur in the implementation.
The clean dataset was eventually reduced to an entirety of 936 entry rows for further evaluation use.

7.1 Speed: Mean and Standard Deviation

With the clean dataset divided into the two independent variables: visual feedback type and selection condition, the first step is to calculate the actual speed values and afterwards determine mean and standard deviation numbers. For the speed count, an if-condition is introduced, in which only the rows containing matches are taken into consideration when calculating and displaying the values.
The mean and standard deviation determination is accomplished in two parts, which will be explained hereinafter.

7.1.1 First Attempt: Initial Mean and Standard Deviation

For each subset filtered we will take a look at the mean and standard deviation values, which are determined using the AVERAGE- and STDEV-function provided by Google Sheets. Calculated values are further rounded to the nearest cent and are displayed in milliseconds. The results are presented in table 7.1.1, distinguishing between the two independent variables feedback type and selection condition.

independent variable	subset	mean	standard deviation
feedback type	NO	12100.23ms	12401.76ms
	NORMAL	10204.65ms	13423.21ms
	GRADUAL	12710.51ms	14780.42ms
selection condition	SAME	12852.90ms	14765.98ms
	OPPOSITE	12485.02ms	15236.41ms
	RESPECTIVE	9716.10ms	10341.11ms

Table 7.1: Initial Attempt: Speed Means and Standard Deviations according to Subsets

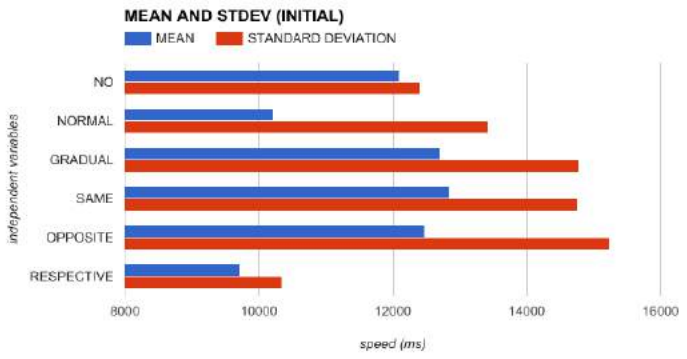


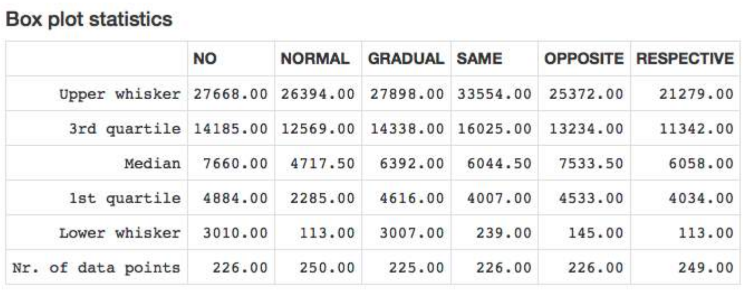
Figure 7.2: Comparable visual illustration of the mean and standard deviation values from table 7.1.1

For a better visualisation and comparison of the calculated means and standard deviations regarding selection speed, the figure 7.2 gives a graphical illustration of the above listed values.
If we compare the average selection speed values, the distribution hints towards normal visual feedback, describing instant feedback given upon selection, registering a faster average selection speed than particpants not receiving any type of visual feedback or getting gradual one. Normal visual feedback supposedly gives a faster overall selection time which preceeds the other two types by approximately 2000ms.
Another interesting point is that selection tasks which gave users gradual feedback were deemed completed taking more time in average than those offering no visual feedback, at a difference

of approximately 600ms, thus supporting the assumption that normal visual feedback tasks were completed at the fastest, with an average speed of 10204.65ms, followed by selection tasks resulting in no visual feedback with an average of 12100.23ms.
The supposedly slowest selection tasks gave users gradual visual feedback, at an average selection speed of 12710.51ms. Albeit these observations, at this point we cannot distinctively say that one feedback type resulted in objects constantly being selected faster than another.
As for the subsets divided according to the different tasks, we notice that tasks requiring the pursuit of the object closest to each user, i.e. their respective objects, were completed at a seemingly faster pace than tasks requiring users to look at objects opposite of them or those asking users to follow the same moving circles.
In this case looking at the objects closest to the user resulted in an average selection speed of 9716.10ms, which is about 3000ms faster than the selection speed acquired with users having to follow the same or their opposite objects. The latter two tasks only had around 400ms of average selection speed difference, hence placing having to look at the user's opposite object at reportedly second fastest with an average selection speed of 12485.02ms.
Finally the requirement to follow the same object was the allegedly slowest in average selection speed with a value of 12852.90ms. Still since this only gives an insight into the average selection speed of 28 participants, we cannot presume that one task was faster than the others in getting the right object and eye movement match.
These statements are fortified if we further take into account the calculated standard deviations, which at first glance seem considerably high, often even higher than the calculated mean value. The standard deviation is considered a means to tell us how spread the values of a specific given dataset are. So, having a larger standard deviation value tells us, that the values calculated are relatively widely spread apart from the mean value, thus there is an existence of outlier, which will have to be detected in a second attempt. Taking a further look at the standard deviation values, the table 7.1.1 as well as figure 7.2 suggest that gradual visual feedback produces more widely spread values in terms of selection speed with a standard deviation of 14780.42ms, followed by normal visual feedback with a standard deviation value of 13423.21ms and no visual feedback with 12401.76ms. This could explain the assumption that allegedly gradual visual feedback results in object selections occurring at a slower pace than normal or no visual feedback.
For the selection condition in regard to the standard deviation values produced, the selection tasks based on selecting the opposite object seem to have the highest standard deviation value at 15236.41ms whereas same object selections have a standard deviation value of 14765.98ms and opposite ones account to a value of 10341.11ms. As mean and standard deviation are not robust against outlier and tend to show noise values immediately by increasing, the following attempt will start with detecting and eliminating outlier.

7.1.2 Second Attempt: Eliminating Outlier

For the second attempt in determining the mean and standard deviation values, first of all outlier should be detected. For this purpose the subsets were plotted using two approaches which will be outlined in the following.
The first approach to find outlier values makes use of median and IQR calculations. This method helps in overcoming the prior mentioned susceptibility of mean and standard deviation towards outlier by setting an upper and lower limit for values which are to be removed from the dataset, as they are considered outlier. For that matter we calculate the needed values described in figure 7.3, using BoxPlotR, an online web-tool which provides all the calculations behind-the-scenes and displays the graphics showing the amount of outlier determined [62]. In this case we only need to import the speed data divided by the independent variables feedback type and selection condition, which results in six columns containing speed values. Important for a box plot is in general the median, around which the values are situated, the lower and upper quartile, which mark the upper



Box plot statistics

	NO	NORMAL	GRADUAL	SAME	OPPOSITE	RESPECTIVE
Upper whisker	27668.00	26394.00	27898.00	33554.00	25372.00	21279.00
3rd quartile	14185.00	12569.00	14338.00	16025.00	13234.00	11342.00
Median	7660.00	4717.50	6392.00	6044.50	7533.50	6058.00
1st quartile	4884.00	2285.00	4616.00	4007.00	4533.00	4034.00
Lower whisker	3010.00	113.00	3007.00	239.00	145.00	113.00
Nr. of data points	226.00	250.00	225.00	226.00	226.00	249.00

Figure 7.3: Calculations of values necessary for graphing box plots

and lower threshold of the box and additionally the lower and upper whisker which illustrate a the outer threshold for outlier detection [2]. The tool further allows for outline modifications and moving the placing of the plot, which finally resulted in the box plot shown in figure 7.4.

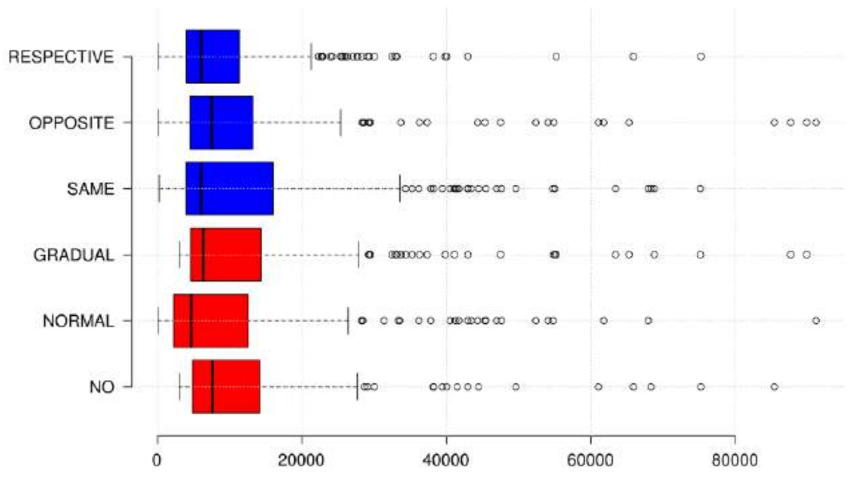


Figure 7.4: Box plot diagramm containing speed values according according to independent variables feedback type and selection condition

If we take a closer look at the results illustrated in figure 7.4 we can initially observe a high amount of outlier detected by the plotting algorithm, many of which, such as seen for the respective object speed boxplot, lie close to each other. Taking the calculated upper whisker and lower whisker as the upper and lower final threshold for assumed outlier and eventually removing those from the original subset gives the following results: For no visual feedback the subset decreased from an initial size count of 226 to 208, normal visual feedback supposedly contained 25 outlier which were removed and gradual visual feedback fell from 225 sample entries to 199. As for the second independent variable selection condition, the re-calculations showed a decrease of sample entries from 226 to 199 for speed on selections of the same object, from 226 to 203 for selections on the opposite object and finally looking at the respective object delivered a decrease of speed entries from 249 to 219.
Thus, outlier detection using box plotting according to median and IQR values suggests the removal of approximately 69 entries from the original dataset to keep them from interfering with later statistical significance tests.
The second approach utilises an upper threshold and makes use of the condition that the current speed value is evidently an outlier in case it exceeds the given mean for the respective condition added to three times the standard deviation count of that independent variable (IV), i.e. we have to check speed subsets based on the equation 3. We keep all the values below that threshold. This

outlier detection method is straightforward but tries to keep a majority of values and results in the removal of a total of approximately 18 outlier.

$$(avg(speed) > mean(IV) + 3 * stdev(IV)) = OUTLIER \tag{3}$$

These two approaches both result in outlier detection, but we cannot assume one method is better than the other, as both bring advantages and disadvantages. While using box plotting as means of outlier detection delivers a more homogenous dataset and thus ensures consistency throughout the speed values, it removes almost all the digressions throughout the dataset, i.e. it is too strict. A dataset with a widely spread value ocurrence profits from using the second approach as it offers more leeway for higher values to be counted as part of the set. On the other hand the existence of undetected assumed outlier in a dataset can lead to problems during statistical significance evaluations later on.
For the course of the following evaluation we will use the data retrieved after using the second approach of outlier detection and removal. After establishing this, it is interesting to redo means and standard deviations calculations to see whether they differ from the initial attempt. Table 7.1.2 shows that there is a slight decrease in not only the mean speed values but also the according standard deviations. While the supposed ranking of speed remains the same with normal visual feedback and respective object selections allegedly resulting in a faster average selection speed, the standard deviation values are not as high as during the first attempt, which hints at a more homogenous dataset.

independent variable	subset	mean	standard deviation
feedback type	NO	10587.72ms	8249.46ms
	NORMAL	8890.05ms	10387.42ms
	GRADUAL	11003.95ms	10572.97ms
selection condition	SAME	11589.48ms	12255.33ms
	OPPOSITE	10409.14ms	9719.02ms
	RESPECTIVE	8898.23ms	8031.73ms

Table 7.2: Second Attempt: Speed Means and Standard Deviations according to Subsets

7.2 Error Rate: Mean and Standard Deviation

Having established which outlier to remove from the clean dataset the next step involves finding mean and standard deviation values for the error rate. To quickly recap: errors were logged as such whenever a user attempts to manually reset the selection made by the system. Before removing the outlier rows, we perform the calculations on the entirety of the clean dataset to see whether there is a visible difference between having outlier removed or not. The table 7.2 shows a direct comparison of average error rate prior to outlier removal according to speed and afterwards.

independent variable	subset	mean	SD	mean (no)	SD (no)
feedback type	NO	0.18	0.44	0.26	1.00
	NORMAL	0.26	0.61	0.21	0.50
	GRADUAL	0.34	0.87	0.29	0.77
selection condition	SAME	0.24	0.56	0.20	0.50
	OPPOSITE	0.32	0.80	0.25	0.66
	RESPECTIVE	0.22	0.61	0.30	1.05

Table 7.3: Comparison error rate before and after outlier removal

Taking a look at table 7.2 we discover that the average error count and the resulting standard deviation value for no visual feedback based selections, seems to increase rather than decreasing after outlier removal, from an average of 0.18 errors with a standard deviation of 0.44 to a mean of 0.26 and standard deviation of 1.00. We keep in mind that the outlier are detected using the second approach described in the part on outlier removal. But we use the already removed speed values as reference. The same trend can be observed for tasks based on selections of the user's respective objects which allegedly produced an average of 0.22 errors with a standard deviation of 0.61 prior to outlier removal and a mean of 0.30 and a standard deviation of 1.05 afterwards.
On a first glance, the table suggests that prior to the process of detecting and removing outlier values, no visual feedback has participants reaching for the reset button less than normal feedback with mean of 0.26 and standard deviation of 0.61 at second least errors, and finally gradual visual feedback which has users reset at an average of 0.34 and standard deviation of 0.87, being allegedly the most prone to users resetting their selections. As for the second independent variable, the selection condition, prior to outlier removal, selections involving the object closest to each user, supposedly produce less errors in average, followed by interactions with the same object at an average of 0.24 and a standard deviation of 0.56, and lastly opposite selections with a mean of 0.32 and a standard deviation of 0.25. This signals that it seems harder for the system to indicate the match when selecting objects that are closer to the respective user's partner.
After outlier removal based on speed however, no visual feedback and respective object based selections seem to produce more errors than their counterparts, which overall result in less errors than beforehand.
One possible explanation for this could be that because of the outlier removal according to speed value there is a slight interference of the average values concerning errors, as speed and error rate at this point are not linear dependent from each other, i.e. if the speed increases the error rate does not increase as well. The verdict could be that a particularly fast selection has the specific participant resetting their selections more often than another object being selected at a slower selection time. Another reason for the switch from being the one condition allegedly resulting in the less number of errors to the most numbers for no visual feedback in particular could be that the system sometimes took longer to respond to users' gaze behaviour during the study, which led to users getting impatient and thus pushing their reset button.
In general we cannot assume that because of these mean and standard deviation calculations, one feedback type or selection condition resulted in less errors made, as the choice finally lay within the user themselves. In order to completely be able to either reject or accept this hypothesis, we need to perform significance tests first.

7.3 Distraction Count: Mean and Standard Deviation

In order to keep count of the times one user found themselves distracted by the feedback provided for the other user, we assume that there is a match between one user's looked at objects and their partner's expected objects during the selection task. We had to post-process the entirety of the clean datasheet and manually check for these types of fits. To facilitate this process, we took only the entries containing a selection process where prior to a user's match between the object they were supposed to select and their actual selection, other objects had been selected first. Afterwards we divided the selection processes into the respective independent variables feedback type and selection condition. This left us with 432 rows of data for feedback type to validate according to distraction counts. For this matter we had one window with the selection tasks listed and checked according to whether it was the left user or the right one making the current selection and if they had a match with their partner's expected object.
After completing the distraction count, we further divided the entries that marked a distraction according to the selection condition, to get the initial six columns for further evaluation use. Also, there are three prior assumed conditions for counting distractions: First of all for no visual feed-

independent variable	subset	mean	standard deviation
feedback type	NO	0	0
	NORMAL	0.13	0.33
	GRADUAL	0.15	0.35
selection condition	SAME	0	0
	OPPOSITE	0.14	0.34
	RESPECTIVE	0.13	0.33

Table 7.4: Distraction means and standard deviation according to independent variables

back we assume that no distractions are presented to the user, therefore each count found in this category is replaced by the value 0. Furthermore, for same object selection tasks, we also assume that no visual distractions happen, as both users are expected to get the same visual feedback on the same object. Additionally, only one distraction can occur per selection task, as we seek matches to the partner's expected object, rather than all of their selections as well.
If we have a look at the calculated mean and standard deviation values, we can see that as expected given the assumptions prior to counting distractions, both selections giving no visual feedback and those resulting in having to select the same object, return a mean of 0 and and according standard deviation of 0. Table 7.3 shows a summary of the calculated mean and standard deviation values.
As for the rest of the independent variables, distraction counts seemingly with a similar standard deviation ranging from 0.33 to 0.35, suggests that normal visual feedback results in a minimally less average distraction count at 0.13 than gradual feedback at 0.15. For the selection condition having to select respective objects allegedly led to a marginal smaller number of average distraction counts, 0.13, as opposed to having to look at the user's opposite object, at 0.14 average distractions.
The occurrence of these numbers could be explained in the following way: For normal feedback selections it makes sense for the average distraction count to appear lower than for gradual feedback, as it is codependent on the speed with which the object is selected, therefore we could speak of a potential codependence, which cannot be taken as certain at this point of the evaluation process. On the other hand, having to look at the objects closest to oneself logically seems to have the lower distraction count, as the user tends to not recognise what is occurring at the other end of the screen, if they fixate on objects in their immediate reach, whereas we can be distracted more by something appearing in our direct periphery.
However, seeing as the values do not differ too much between the independent variables, we cannot assume that calculating means and standard deviations for distraction counts evaluate these types of variables in the most sufficient way, as they do not show any significant differences throughout the respective independent variables. For that matter we still need to perform significance tests later on.

7.4 Comparing User Estimations with System Logs

Since we spoke about means and standard deviations in the previous sections, it makes sense to illustrate graphically a direct comparison between what the participants for our study themselves thought about how fast selections were made, and how many times they reset their selection. Since there was no estimate on numbers regarding the times the user felt distracted by their partner's selection, we will not include this comparison at this point. Hereinafter we have a direct comparison of the mean and standard deviation values according to the calculations made in prior chapters and the participants' input on their estimations on selection speed and error rate for the three feedback types, as the questionnaire only had questions divided into three parts based on the visual feedback types. Figure 7.5 illustrates the error rate comparison.

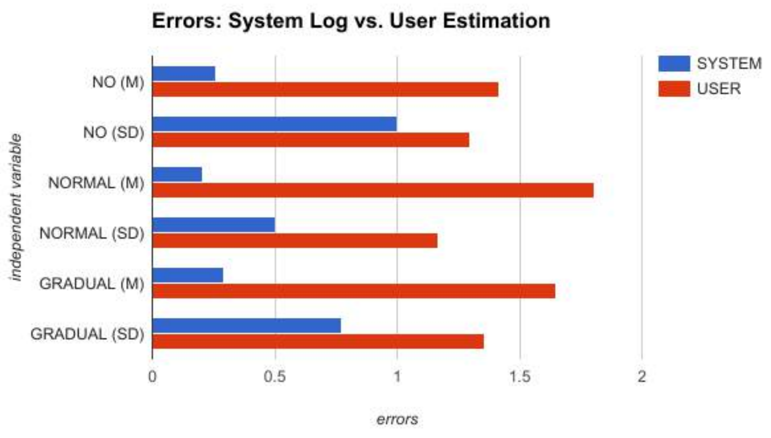


Figure 7.5: Mean and standard deviation of error rate based on system calculation and user estimation

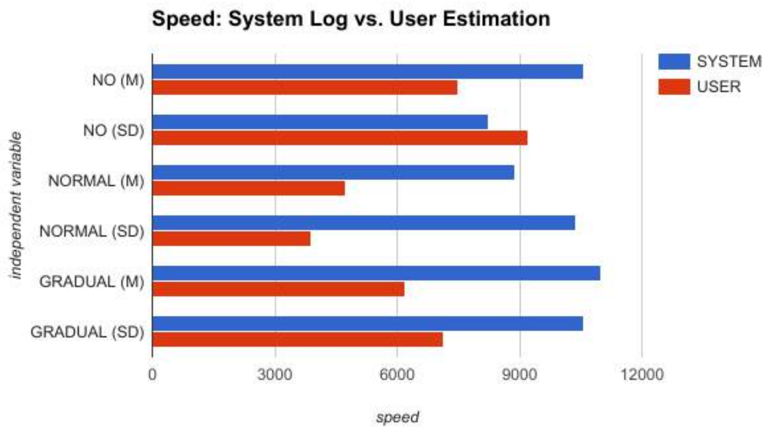


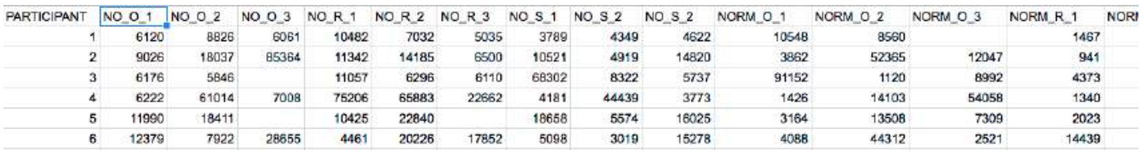
Figure 7.6: Mean and standard deviation of speed values based on system calculation and user estimation

In general participants reckon they have been pushing the reset button throughout interaction more often than the system actually logs. If we separate mean and standard deviation comparisons, according to user estimation and system calculation, we can see, that users tend to think that the system takes a long time to recognise their selection, thus they feel the need to reset their selection in order for them the system to improve, i.e. a better selection starting point resulting in errors being tracked. In average participants feel that no visual feedback produces less errors at an average error rate of 1.4 with a standard deviation of 1.23, which is considerably less than their estimate for the average error rate for normal visual feedback at 1.81 with a standard deviation of 1.17. Gradual feedback according to users produces a medium amount of errors with a mean of 1.65 and a standard deviation of 1.35. If we now compare these values to the calculated system mean and standard deviations, apart from generally higher estimations, users also find that normal visual feedback produces the most errors, which the system does not illustrate, whereas no visual feedback produces the least errors, which makes sense, since the users do not see whether they select the right object until the sound resonates, at which point the task is already finished. A reason for the high difference between estimation and system log could be the fact that users opt to not push

the reset button as often as they think, because the selection process happens relatively fast.
For speed averages and standard deviations, figure 7.6 shows that the opposite happens to user estimations in terms of speed values for the three feedback types. Participants perceive normal feedback as the fastest selection tasks with an average speed of 4727.78ms and standard deviation of 3905.43ms. The allegedly second fastest selections are produced during gradual feedback displays at an average speed of 6182.22ms and a standard deviation of 7133.54ms, whereas no visual feedback is perceived as the slowest selection task with a mean of 7494.26ms and a standard deviation of 9201.29ms. This observation made by participants generally correlates with the calculations and logs made by the system, although gradual feedback in this case supposedly takes longer to achieve than tasks resulting in no visual feedback, hinting at normal visual feedback being the fastest achieved type of feedback.
During the course of the past sections we spoke about hints and assumptions made based on mean and standard deviation calculations, but these assumptions stay what they are, since we cannot take observations as a given to accept any hypothesis as this point. The next step is to proceed with significance tests.

7.5 ANOVA Significance Tests

Analysis of Variance tests, short ANOVA, aim to help evaluation in terms of accepting or rejecting prior formulated hypotheses. These types of statistical techniques are testing the hypotheses. Since the user study at hand involves repeated measuring of selections tasks, we will step-by-step analyse ANOVA tests in this section in regard to speed, error rate and distraction count. All ANOVA testing is done using IBM's SPSS software.
Prior to conducting the tests, we rearranged the dataset to fit the repeated measure ANOVA test purposes: Each row contained all the selections made per participant, thus resulting in 28 rows, as we had 14 groups at 2 participants each. Furthermore, we divided the values along the feedback type, having three groups each containing no visual feedback, normal visual feedback and gradual visual feedback. In addition to that, selections were further divided by the selection conditions opposite, respective and same and finally the selections repetitions were divided by number, i.e. first selection, second or third, resulting in 28 columns, as we had 27 selections to be made and one column containing the participant ID. In general we had the following layout prepared, which is illustrated in figure 7.7.

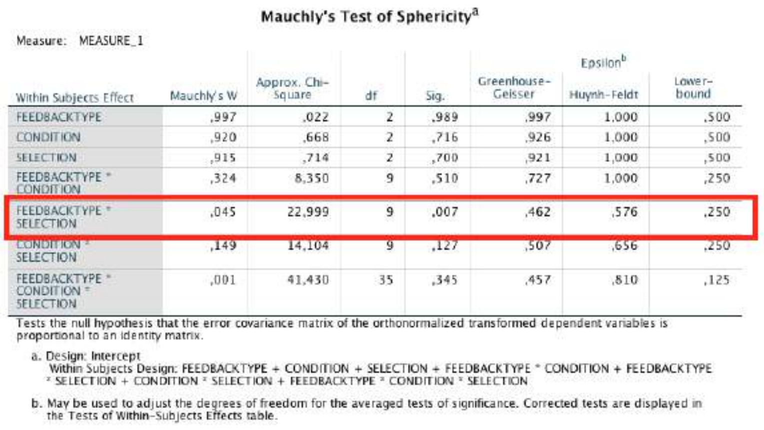


PARTICIPANT	NO_O_1	NO_O_2	NO_O_3	NO_R_1	NO_R_2	NO_R_3	NO_S_1	NO_S_2	NO_S_2	NORM_O_1	NORM_O_2	NORM_O_3	NORM_R_1	NOR
1	6120	8826	6061	10482	7032	5035	3789	4349	4622	10548	8560		1467	
2	9026	18037	85364	11342	14185	6500	10521	4919	14820	3862	52365	12047	941	
3	6176	5846		11057	6296	6110	68302	8322	5737	91152	1120	8992	4373	
4	6222	61014	7008	75206	65883	22662	4181	44439	3773	1426	14103	54058	1340	
5	11990	18411		10425	22840		18658	5574	16025	3164	13508	7309	2023	
6	12379	7922	28655	4461	20226	17852	5098	3019	15278	4088	44312	2521	14439	

Figure 7.7: Layout example for speed values to import into SPSS

We abbreviated the label for the columns according to the feedback type (NO, NORM and GRA), then the selection condition (O, R and S) and finally the number of selection (1, 2 and 3). If specific tasks were not fully tracked by the system, we left the respective cell blank. The sheets for error rate and distractions were created in an equal manner.
For ANOVA tests on speed and error rate we further created a duplicate of the existing sheet and removed the outlier detected with the second approach in section 7.1.2. We wanted to see whether outlier removal would help strengthen significance or not. In the following we will take a look at the three dependent variables speed, error and distraction to see whether there are significant results in any way.

7.5.1 Speed



Mauchly's Test of Sphericity[a]

Measure: MEASURE_1

					Epsilon[b]		
Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Lower-bound
FEEDBACKTYPE	,997	,022	2	,989	,997	1,000	,500
CONDITION	,920	,668	2	,716	,926	1,000	,500
SELECTION	,915	,714	2	,700	,921	1,000	,500
FEEDBACKTYPE * CONDITION	,324	8,350	9	,510	,727	1,000	,250
FEEDBACKTYPE * SELECTION	,045	22,999	9	,007	,462	,576	,250
CONDITION * SELECTION	,149	14,104	9	,127	,507	,656	,250
FEEDBACKTYPE * CONDITION * SELECTION	,001	41,430	35	,345	,457	,810	,125

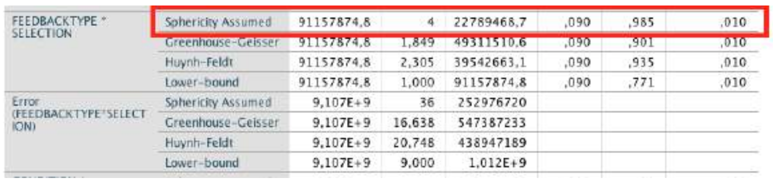
Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. Design: Intercept
Within Subjects Design: FEEDBACKTYPE + CONDITION + SELECTION + FEEDBACKTYPE * CONDITION + FEEDBACKTYPE * SELECTION + CONDITION * SELECTION + FEEDBACKTYPE * CONDITION * SELECTION

b. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.

Figure 7.8: Mauchly's test of sphericity for speed averages

We conduct a repeated measure ANOVA test using the help of a tutorial on SPSS [1]. For speed means comparison we can say that the test does not result in significance being found, except for feedback type*selection, which is illustrated in figure 7.8 at a significance of P = 0.007, which lets us assume sphericity. According to Mauchly's test of sphericity, having the product between feedback type and selection tested on significance could result in a significant difference of speed averages. This thought unfortunately is not verified if we take a look at the table containing within-subject effects shown in figure 7.9, where the significance value is 0.090 which is higher than P = 0.05.



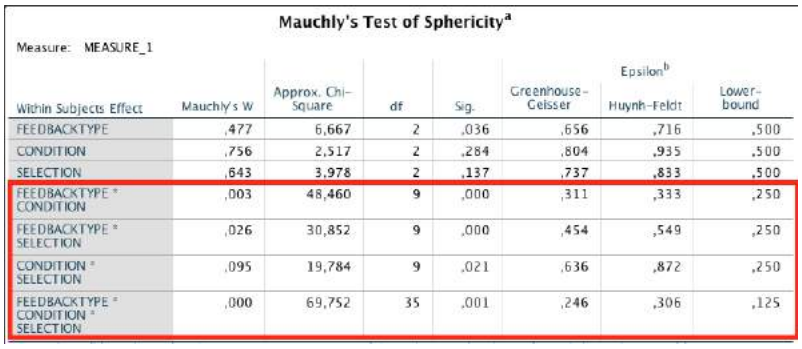
FEEDBACKTYPE * SELECTION	Sphericity Assumed	91157874,8	4	22789468,7	,090	,985	,010
	Greenhouse-Geisser	91157874,8	1,849	49311510,6	,090	,901	,010
	Huynh-Feldt	91157874,8	2,305	39542663,1	,090	,935	,010
	Lower-bound	91157874,8	1,000	91157874,8	,090	,771	,010
Error (FEEDBACKTYPE*SELECTION)	Sphericity Assumed	9,107E+9	36	252976720			
	Greenhouse-Geisser	9,107E+9	16,638	547387233			
	Huynh-Feldt	9,107E+9	20,748	438947189			
	Lower-bound	9,107E+9	9,000	1,012E+9			

Figure 7.9: Testing within-subject effects for significance

According to this table, as we have sphericity confirmed, there is no statistical significance visible. Testing on *Mauchly's test of sphericity*, *Greenhouse-Geisser*, *Huynh-Feldt* and *Lower-Bound* does not provide significance values lower than P = 0.05, which therefore result in us not being able to statistically assume that there is a difference between the averages concerning selection speed for the different independent variables.
To see whether removing outlier results in a significant change, we compare the results generated for speed averages without and with outlier, especially to see whether for the above conducted test for the feedback type and selection product would result in a significance. While removing outlier with boxplot does not deliver any results, removing outlier using the second approach of removing speed values that are higher than the equation given in section 7.1.2, results in the same results as testing with the dataset still containing all outlier.
In conclusion to that, we cannot accept the hypothesis and have to reject it in favour of the null hypothesis. However, we can still hold onto the tendency of speed averages for normal visual feedback and respective objes determined via mean and standard deviation calculations, to be lower as described earlier.

7.5.2 Error

If we regard error averages for our ANOVA repeated measure test, we can see that in this case, the test generates more possible significance values using Mauchly. Figure 7.10 shows that for products between independent variables sphericity is assumed and we can look at the table containing within-subject effects. Looking at table delivers no significant results for the products between feedback type and condition, feedback type and selection, condition and selection and the multi-product between feedback type, condition and selection does not contain significant differences between error averages as hoped after having sphericity assumed.



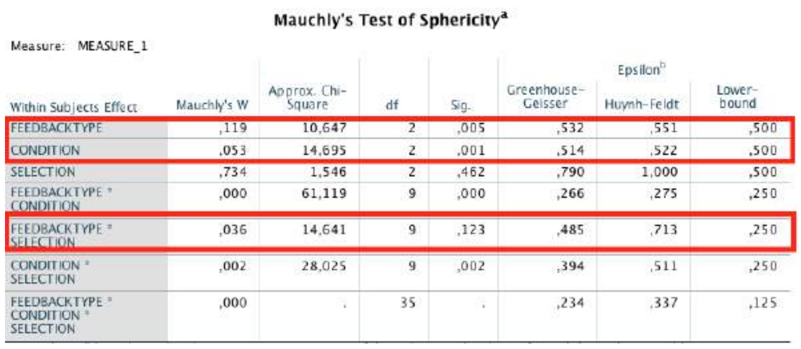
Mauchly's Test of Sphericity[a]

Measure: MEASURE_1

					Epsilon[b]		
Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Lower-bound
FEEDBACKTYPE	,477	6,667	2	,036	,656	,716	,500
CONDITION	,756	2,517	2	,284	,804	,935	,500
SELECTION	,643	3,978	2	,137	,737	,833	,500
FEEDBACKTYPE * CONDITION	,003	48,460	9	,000	,311	,333	,250
FEEDBACKTYPE * SELECTION	,026	30,852	9	,000	,454	,549	,250
CONDITION * SELECTION	,095	19,784	9	,021	,636	,872	,250
FEEDBACKTYPE * CONDITION * SELECTION	,000	69,752	35	,001	,246	,306	,125

Figure 7.10: Mauchly's test of sphericity for errors averages

To take into consideration outlier removal as well, we try ANOVA on the same dataset with outlier removed in regard to speed values. Figure 7.11 shows that there is a change as for variables being tested resulting in a significance in the column sphericity, which lets us believe that there might be a significant difference in average errors. Feedback type and selection condition have a significance of P = 0.005 and P = 0.001 which assumes sphericity, whereas feedback time * selection no longer is in the sphericity assumption list. If we look at the table containing within-subject effects however, we cannot spot any significant values which is roughly the same result we get for pre-outlier detection.



Mauchly's Test of Sphericity[a]

Measure: MEASURE_1

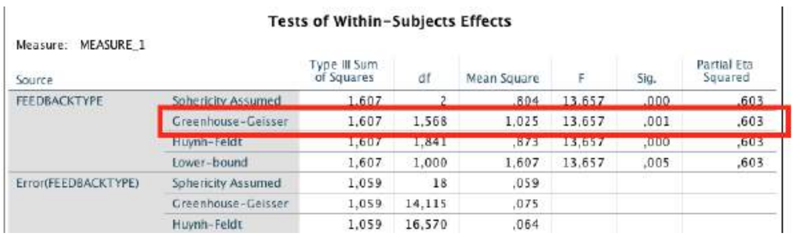
					Epsilon[b]		
Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Lower-bound
FEEDBACKTYPE	,119	10,647	2	,005	,532	,551	,500
CONDITION	,053	14,695	2	,001	,514	,522	,500
SELECTION	,734	1,546	2	,462	,790	1,000	,500
FEEDBACKTYPE * CONDITION	,000	61,119	9	,000	,266	,275	,250
FEEDBACKTYPE * SELECTION	,036	14,641	9	,123	,485	,713	,250
CONDITION * SELECTION	,002	28,025	9	,002	,394	,511	,250
FEEDBACKTYPE * CONDITION * SELECTION	,000	.	35	.	,234	,337	,125

Figure 7.11: Mauchly's test of sphericity for errors averages

In conclusion to that, we cannot accept the hypothesis and have to reject it in favour of the null hypothesis, as ANOVA did not deliver significant differences throughout the several independent variables. However, we can still assume the trend determined via mean and standard deviation calculations of error values as described earlier.

7.5.3 Distraction

As for distraction count sorted by feedback type, selection condition and selection made, we previously agreed that selections resulting in the same object being selected and selections receiving no visual feedback were considered to have 0 distractions for every participant. Still, conducting a repeated measure ANOVA based on these assumptions delivers interesting results.
After generating the output for repeated measure ANOVA, we first need to look whether we can assume sphericity for the following evaluation. *Mauchly's test of sphericity* in this case does not deliver a significant value, thus the assumption is violated, which results in further analysis involving looking at the values in the *Greenhouse-Geisser* rows instead. For *tests within-subject effects* we are interested in rows using the Greenhouse-Geisser correction which have a significance level of below 0.05. At first glance all of the respective subjects in figure 7.12, we notice that there is an overall significance detected. We can state that when using an ANOVA repeated measures test, with a Greenhouse-Geisser correction, the mean scores for distraction counts are statistically significantly different between feedback types (F(1.568,14,115) = 13.657, p < 0.005), between selection conditions (F(1.691,15.223) = 7.059, P < 0.01) and selection tasks (F(1.492,13.425) = 37.406, P < 0.0005). Now that we established the existence of statistical significance of difference between the independent variables, we however, do not know at present, which variable is affected in which way, which is why we need to use post hoc tests to determine which specific means of distraction counts differ.



Tests of Within-Subjects Effects

Measure: MEASURE_1

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
FEEDBACKTYPE	Sphericity Assumed	1,607	2	,804	13,657	,000	,603
	Greenhouse-Geisser	1,607	1,568	1,025	13,657	,001	,603
	Huynh-Feldt	1,607	1,841	,873	13,657	,000	,603
	Lower-bound	1,607	1,000	1,607	13,657	,005	,603
Error(FEEDBACKTYPE)	Sphericity Assumed	1,059	18	,059			
	Greenhouse-Geisser	1,059	14,115	,075			
	Huynh-Feldt	1,059	16,570	,064			

Figure 7.12: Results of within-subject tests for distractions based on feedback type, selection condition and selection count

As we now have overall statistically significant difference of means, we will take a look at the pairwise comparison for each independent variable as well as pairing of variables. For the pairwise comparison table of feedback type, we can say that comparing average distraction counts of no visual feedback to both types of visual feedback resulted in a significant difference of means between them, at P = 0.006 for comparison of no visual feedback and normal visual feedback, and 0.009 for no visual feedback and gradual visual feedback, which is both times lower than 0.05. So we can statistically say that having no visual feedback resulted in less average distractions being recognised, than selections giving either of the visual feedback types. However, when we regard the comparison of average distraction count between normal and gradual visual feedback, there is no statistical difference noticable, as the significance value remains higher than 0.05. In this case we cannot say anything to determine that one or the other visual feedback type results in less or more distractions. This is not the most interesting observation, as we assumed that no visual feedback would result in no distractions, before counting distraction numbers per selection, so we already expect such a result.
As for selection condition, same, respective and opposite, we can also assume that same object interactions would result in no distractions by the partner's feedback, therefore we also expect the results given: Interaction on the same object produces significantly less distractions in average, with a significance value of 0.036 than opposite objects. Furthermore same object interactions produce significantly less distractions than selections involving the respective objects with a value of 0.008. As for the comparison between opposite and respective object selections in terms of distraction count, we cannot find any significant difference, therefore we cannot assume that either

of the two selection conditions produce more or less distractions for users.
For the current repetition of selection, which marks which of the three runs was currently tracked, it can be said that these were arbitrarily ordered by us, i.e. that the first repetition was almost always filled whereas the third repetition would sometimes be missing values, as the system did not correctly track the selection. Also we are counting selections that provide no visual feedback and selections that involve interacting with the same object into our means comparison as well, which results in the first task repetition being significantly lower in distraction count at P = 0.001 than the second, and P = 0.000 for the comparison between the first and the third selection repetition. There is no significant difference between the second and the third task repetition in terms of distraction average.
So overall, the ANOVA shows us what we expect and assume for the feedback type, that having no visual feedback results in statistically significantly less distractions than having feedback, as well as selecting the same object resulting in less distractions than opposite or respective objects, as well as the first repetition producing less distractions than the third or second.

7.6 Questionnaire Evaluation

In this part we learn more about our participants and gauge their opinions on the feedback giving system and interaction with the interface by analysing the 5-point Likert scale options ranging from 1 (strongly disagree) to 5 (strongly agree), chosen from the questionnaire, as well as their sentiment on which type of feedback and selection they prefer and what they would improve or change on the given three types of feedback and selection conditions in written and verbal form.

7.6.1 Demographic Evaluation

First of all it is important to analyse what type of demographics the participants represent for our study. The study in its entirety had 28 participants divided into 14 groups of two users each time and can be described as follows: The 28 participants consisted of a gender distribution of 19 male participants and 9 female participants, which resulted in an overall distribution of 67.9 percent to 32.1 percent as can be seen in figure 7.13.

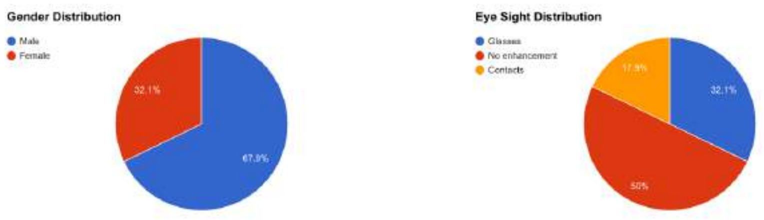


Figure 7.13: (a) gender and (b) eye sight distribution evaluating participant statistics

We also included the options *prefer not to say* and *other* to keep our questionnaire politically correct. The participants' ages ranged between 18 and 32 with an average age of 23.93 and a standard deviation of 3.51. Furthermore, all of the participants were students, albeit not all of them having experiences in gaze interaction prior to this study.
In addition to that, participants proved to have a rather heterogeneous distribution in terms of eye sight statistics, with 50 percent of the participants having no visual correction devices and the other 50 using vision enhancement tools, such as glasses which 9 of the participants wore during the interaction process, thus marking 32.1 percent, and the rest wearing contact lenses, a count of 5 users, which make up 17.9 percent. A visualisation of the eye sight distribution can be found in

figure 7.13. Participants were further asked about their experiences in gaze interaction and their expertise in pursuit interaction prior to the study which was evaluated as shown in 7.14.

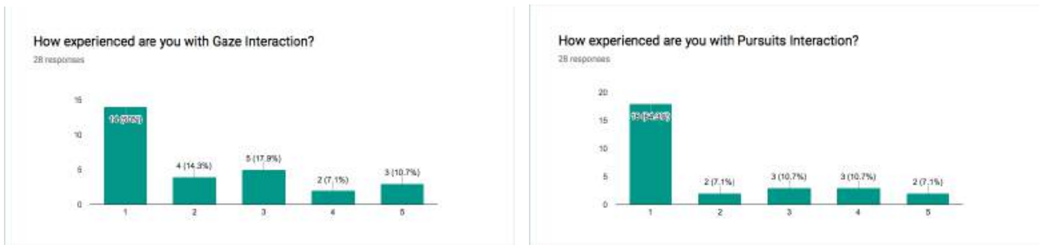


Figure 7.14: (a) experience on gaze and (b) pursuit interaction prior to study, illustrated evaluation Likert scales

The majority of users have not used or had minimal encounters with eye-tracking devices prior to participating and therefore have not heard of pursuit interaction as of yet, In numbers expressed, about 50 percent did not have prior experience whereas 64 percent have not heard of pursuit interaction. The rest is placed ranging from little to medium experience with eye-tracking and pursuit interaction, for eye-tracking being 39.3 percent and pursuit interaction 28.5 percent. The distribution further shows that we had expert participants as well. In average, participants had less expertise in gaze interaction prior to the user study with an average of 2.14 and a standard deviation of 1.41 and similarly less knowledge about gaze pursuits with a mean of 1.89 and a standard deviation of 1.37

7.6.2 No, Normal and Gradual Feedback according to User

After immediate interaction resulting in no, normal and gradual feedback the participants had to fill out the questionnaire according to the feedback type. For all three feedback types the same questions were asked. Participants were required to fill out Likert scales on them agreeing or disagreeing on the following aspects regarding the interaction process: Questions concerning the ease of use and the speed of objects being selected based on the users sentiments. Furthermore they were asked on how comfortable they felt using this specific technique or how confident they were that the system selected the right object according to where they were looking. In addition to that they evaluated whether they felt distracted by the feedback type given at times, among other questions. In these cases we implement Likert scales ranging from 1 to 5 with the lower being strongly disagree and the higher being strongly agree. we opted to include the number 3 to keep the selections from becoming too biased.
To compare the three types it is necessary to evaluate the answers of the questionnaire. As for ease of use and learning the technique at hand, participants in general found it easy to apply gaze pursuit, as it merely involved following a moving object on the screen with our eyes, something that is common in everyday life. Still, albeit the same technique being used every time, participants' answers changed throughout the feedback conditions, resulting in different distributions of Likert scale positions. This might be the case because of difficulties in tracking and matching the object movement with gaze movement for some users, which in turn resulted in them thinking a particular type of feedback was harder to achieve. Participants in general seemed to find normal feedback and gradual visual feedback easier to use than no visual feedback, with an average distribution at 3, 4 and 5, i.e. strongly agreeing on the ease of use of the visual feedback techniques, whereas no visual feedback gave a more equal distribution of opinions, as people were not sure if they performed the right selection, until they received sound feedback. The distribution is hinted at in figure 7.15. As for having to decide whether they found the respective technique easy to learn participants opted for answers in the strongly agree to agree sections, which is expected, since pursuing moving things and people with gaze is a natural interaction technique for Human-

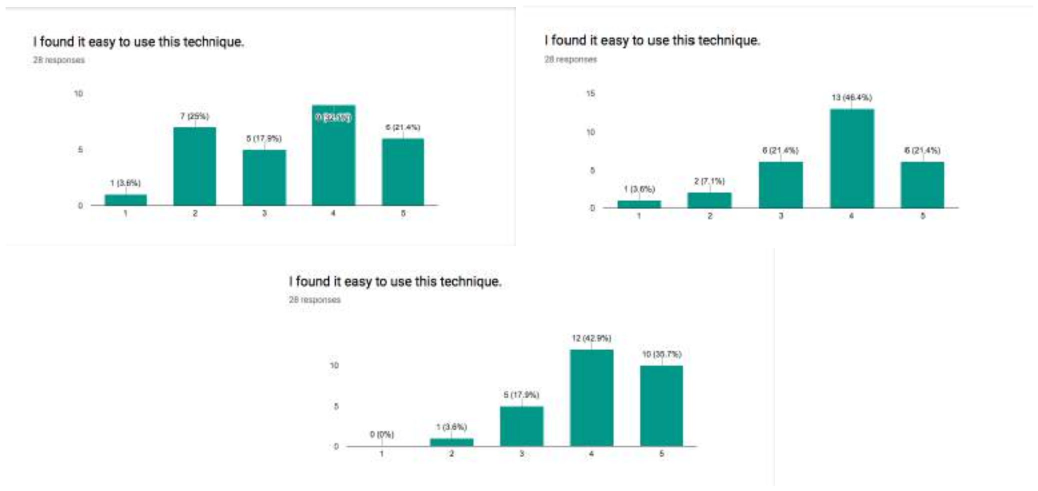


Figure 7.15: Comparison of ease of use for (a) no, (b) normal and (c) gradual visual feedback according to Likert scales

Human-Interaction as well. We tend to look at the things and people we are interacting with most of the times. The answers are similar for the three feedback types.
If we look as the answers regarding sufficiency of feedback, the expected trend was met for the visual feedback types, where users agreed with the statement of getting enough feedback, but the no visual feedback type generated an interesting distribution of opinions. Figure 7.16 shows that the distribution is rather spread, which can be explained by some users giving the opinion that only sound was enough feedback as it was, whereas others preferred having visual feedback, which is understandable when interacting with a computer screen. Also, even though the majority did not like having no visual feedback, there were in the group of participant who preferred only sound over sound and visual.

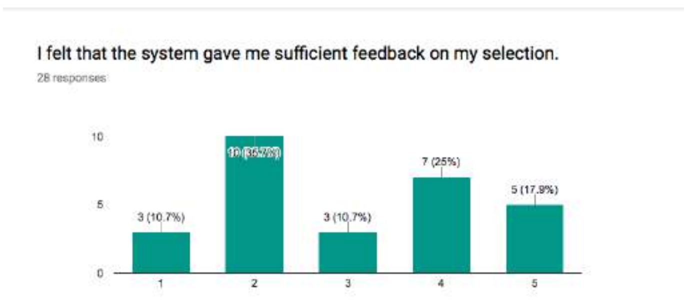


Figure 7.16: Likert scale distribution on no visual feedback giving the sufficient amount of feedback

Furthermore, when asked about confidence in the system in regard to the given feedback, participants generally were not sure whether to strongly agree or disagree and decided to abstain from agreeing or disagreeing too strongly. Still, no feedback resulted in more people opting to disagree with the question whether they felt confident that the system knew where they were looking at all times, which is understandable, as there was no indicator as opposed to during normal and gradual feedback interactions. However, the neutral feelings regarding confidence can be traced back to the fact that the system did not perform without any disruptions.
When asked about whether they felt distracted by the visual feedback, the majority of users either strongly or less strongly disagreed with that statement opting to circle the first or second option, as can be seen in figure 7.17, which shows the comparison of Likert scale distributions for the

three feedback types. However, if we take into consideration the ANOVA tests that stated a significant increase of distraction numbers between no visual feedback and visual feedback, it would have been expected to see that users voted the same way. Also, as explained later on in the section about user opinions, they stated feeling distracted by the visual feedback they or their partner received.

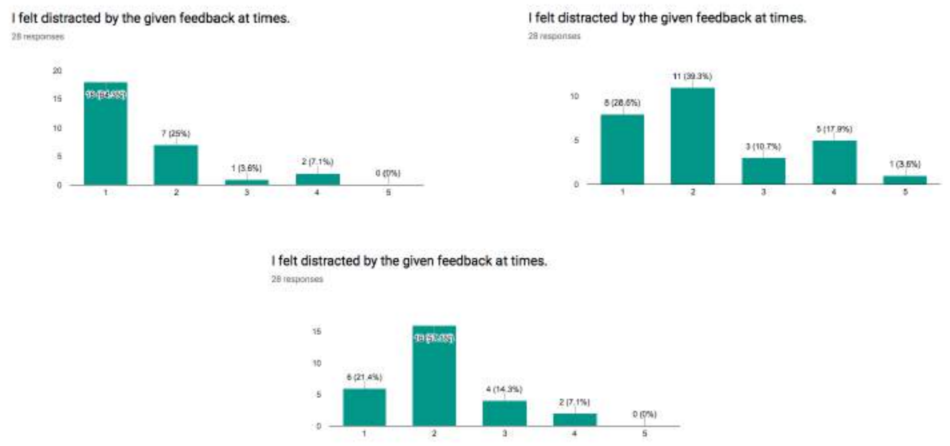


Figure 7.17: Comparison of distraction for (a) no, (b) normal and (c) gradual visual feedback according to Likert scales

Comfort-level-wise participants generally stated that they felt strain in their eyes if they had to look at the screen for a longer period of time. This was mentioned for all three feedback types, because sometimes the system took longer to recognise the selection which turned the interaction more tedious for participants' patience and thus comfort level. Also, having to repeat selection tasks over and over again can be very straining on the eyes.
Finally the participants had to give their opinion on the selection tasks, which they also had to rank. They generally strongly disagreed with feeling awkward while looking at their respective objects, whereas opposite object selections were a bit more spread in agreement or disagreement. Most of the users were either choosing to agree or to disagree, which can be explained by the fact that most of the failed times the system could not give feedback according to selections of opposite nature. If the selection worked and the feedback was given accordingly, users would choose to disagree with awkwardness revolving around opposite interactions. But in case the system did not rightfully give feedback where it was due, users would then lean towards agreeing on feeling awkward while having to select their opposite object. Figure 7.18 illustrates the distribution of agreement for participants on feeling awkward looking at the object closest to them or furthest away from them.

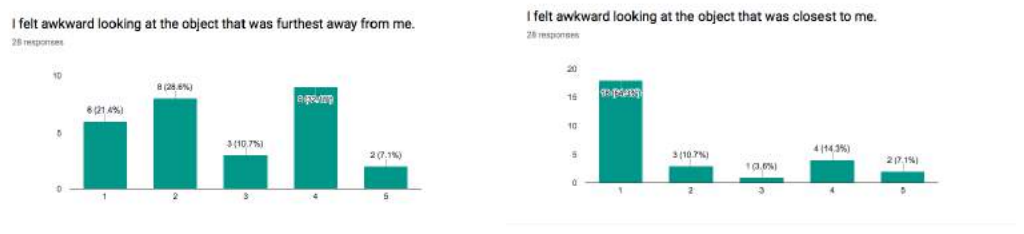


Figure 7.18: (a) Opposite object selections vs. (b) respective object selections

7.6.3 Ranking of Feedback Types

After interaction and opinion giving users were further asked to rank the feedback type they preferred over others, followed by ranking the selection condition and the type of trajectory according to their likability.
Users in general ranked the feedback type as shown in figure 7.19: No visual feedback was not as liked as having visual feedback, with participants mostly preferring gradual visual feedback over normal visual feedback, thus resulting in gradual visual feedback being rank 1, followed by normal visual feedback at rank 2 and no visual feedback at rank 3. When asked why they chose the given rank, participants replied that they were fond of the interactivity coming with gradual visual feedback, as the user had the opportunity to change their gaze behaviour in order to change their selection process before it was finished and acknowledged by the system. No visual feedback was not appreciated as much, as users felt that they were not informed of their selections until the task was completed. Normal visual feedback was generally well liked as well, but there was no visible interactivity according to users.

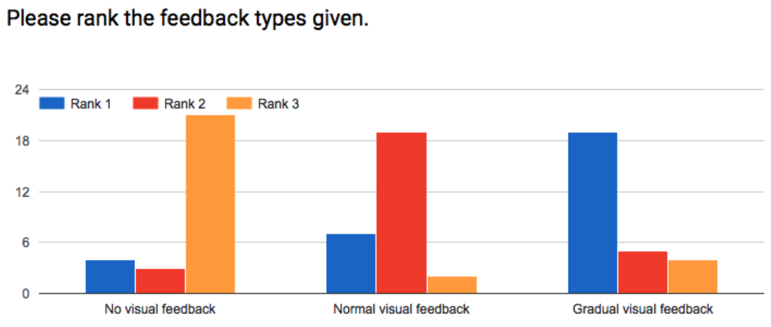


Figure 7.19: Ranking of feedback type according to participants preferences

After asked to rank the feedback types, participants also ranked the selection condition according to their preference. Figure 7.20 suggests that users preferred interaction on the same object over selecting respective objects or opposite objects, with same object selections being ranked at 1 followed by the respective object selections at 2 and opposite at 3. While having to select opposite objects can be understood as being the least preferred in terms of likability, but after analysis of means of speed and error rate and distractions in terms of selection conditions, it would have been expected that users like selections involving the object closest to them more than at least selections involving the same object. The answer to that phenomena could be that users felt a sense of winning if their visual feedback appeared faster than their partners' and having the competition aspect turned interactions more fun according to them.

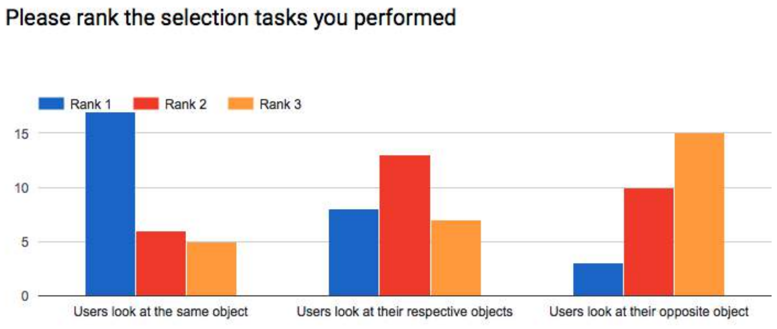


Figure 7.20: Ranking of selection conditions according to participants preferences

Finally we had participants ranking which trajectory movement they preferred, circular moving objects or linear moving circles. Participants liked circular movements more during the interaction. They rather disliked the linear trajectories as compared to the former, but this might be influenced by the fact that the objects closer to the ends of the display were selected less frequently resulting in actual feedback and task completion than the object in the middle as gaze behaviour in that area seemed to be better tracked by the Tobii REX and thus matched.

7.7 User Opinions on Feedback Types

When asked about their opinions on the different feedback types and the overall study, participants stated that they appreciated receiving visual feedback more than not having any visual feedback, because they felt instant gratification that anything was selected at all, thus having feedback on their actions was important to them. Furthermore they stated that they enjoyed having the colour differences when interacting with a partner on the same screen, as it gave them a sense of relief seeing their assigned colour appearing up in any feedback form. In addition to that, they stated the ease of use of the interaction technique gaze and gaze pursuit as they did not have the requirement to actively press a button anymore, but still had the opportunity to select items on the screen.
If we take constructive criticism for each, we can start with analysing people's opinions on getting no visual feedback during selection tasks, although this is not completely feedback-free as they still received a sound when their selection task was completed. Some people liked having a minimum of only one type of feedback (sound in this case), but overall participants stated that there was nothing in particular they enjoyed about this technique and criticized that they did not know what object the system was selecting at what time, which coincides with their confidence level on the system. Furthermore they stated not knowing what the system thought they were looking at until it started playing their assigned sound, was frustrating for them.
The next part was getting instant feedback on selection meaning normal visual feedback. For this method participants generally preferred it over the interaction resulting in no feedback, with comments such as "It was nice to see what you were looking at.". Users also stated that they liked knowing what the system thought they were selecting, as it was interesting to see at times, some even saying that they liked how "subtle yet clear" the visual feedback was and they felt "excited" to get visual feedback. Still, they criticized the accuracy again as well that they felt distracted by their partner's selection feedback and their own visual feedback, i.e. the system did not recognise the right selection at times.
Now as for gradual feedback, participants in general liked the as stated "smoothness" of the feedback process, as well as again, the colours assigned to them. Furthermore some participants liked the fact that having a colour gradation equaled the system asking them whether they were sure they wanted to select a certain object and thus giving them time to re-select. Participants also mentioned that having a sort of process in selection encouraged them to reset their selection more often or have them "adapt their watching behaviour" accordingly. They also were fond of the interactivity of the feedback type. Many also preferred the responsive gradual feedback they got and that they had the "power" to change their selection if they did not want the current selection process to happen. However, on the other hand they wished for a faster selection process, as they felt impatient having to wait until the gradation was fully opaque for their circle to be selected. Also some did not like the specific colour chosen, as blue was hard to spot on the screen, especially when it was not yet fully opaque.
When asked what they would improve or change in regards to the feedback types, many stated to improve the accuracy of the eye-tracker, to maybe use a bigger or public screen to see if interaction would be different there, as the placing of the users in relation to the screen would be different, or changing the trajectories being used to possibly track more matches. A few participants even suggested not using the no visual feedback at all, and a few mentioned that according to the situation it would be nice to be able to switch between gradual and normal visual feedback. However, some

participants preferred the use of only one feedback method (sound or visual), as they said sound would have been sufficient for them.
Furthermore participants wished for changes in the layout, i.e. placing the feedback in the middle of the moving object rather than surrounding it, or placing the half circles laterally to the moving object to better distinguish which user is getting the given feedback, or gradually filling the moving circle with colour for gradual feedback and mixing colours. However, this is only a preference and design question. Other participants stated that adding where the system assumes they are looking would have been helpful to adjust gaze accordingly. Another critique point is that whenever the gaze object match was lost the feedback disappeared immediately, but seeing as this allows for better selectivity, it is better to keep the current system.

7.8 Observations During the Study

During the study we observed participants' interaction process as well as themselves. We could notice that some participants subconsciously began moving their head to follow the object movement better, as they felt that the system would then track their gaze more precisely. When addressed, they did not realise that they were doing so. Furthermore, participant groups where the two users knew each other prior to the study quickly became competitive to see who would finish their selection task more quickly, which could be interesting for later applications involving pursuit feedback research. Also, familiar groups were more quick to discuss their likes and dislikes. In addition to that, although Tobii REX supposedly allows for tracking while wearing glasses, sometimes users with glasses were not rightfully tracked by the hardware. In this case, fortunately they could still interact after removing their eye sight enhancement. It was also tedious for some participants to adjust the tracker angle based on their distance and position towards it, i.e. taller users, albeit sitting, had to be readjusted more often.
Sometimes the eye-tracker would randomly disconnect from the laptop by itself and thus disrupt the interaction process. We assume that either the tracker in question or the laptop have a USB defect for that to happen. Another point was that the system itself did not track gaze on parts of the screen closer to the partner rightfully, resulting in the opposite object not being selected accordingly sometimes. This might be a calibration problem, although both users were calibrated prior to interaction to overcome that problem.

8 Discussion

After evaluating the results of the user study we need to comment on the findings in order to either accept or reject the hypotheses we formulated prior to conducting the study as well as to reflect on what limitations the system still has and classify them. The following chapter will first of all recap the findings as well as basing them on the hypotheses and then secondly list the limitations of our program.

8.1 Acceptance or Rejection of Hypotheses

Having visual feedback proved to have an effect on people getting distracted during their interaction process. No visual feedback logically resulted in less to no distractions by the partner's selections. Furthermore interactions on the same object, i.e. having to select the same object as the partner also statistically significantly proved to result in more distractions being count. Therefore, if we take into consideration the hypotheses formulated in chapter 3, *Concept Develpment*, we can see that the hypothesis stated that getting visual feedback would lead to more distractions than getting no visual feedback, in the first part of the hypothesis. We have a significant difference tracked throughout the feedback types, in particular in comparison of no visual feedback and visual feedback (normal or gradual). To recap below are the first hypothesis and null hypothesis for distractions and errors.

Hypothesis (H1): *Getting visual feedback leads to more distractions or errors amongst users than receiving no visual feedback.*

Null Hypothesis (H1): *Getting visual feedback does not lead to more distractions or errors amongst users than receiving no visual feedback.*

If we look at the hypothesis formulated on distraction count being affected by visual feedback, we can successfully say that for distractions at least, we can accept the hypothesis over its according null hypothesis. As for error rate influenced by visual feedback more than by having no feedback, we unfortunately cannot accept the second part of the hypothesis due to not having statistical significance that there is a difference of average error rate between having and not having visual feedback, although one would think that visual feedback encourages the user to reset their selections more, resulting in more errors being tracked. The explanation for this happening could be that because of system limitations, the user gets impatient and ends up resetting their selection nonetheless. Therefore we have to reject the hypothesis for the part on errors in favour of the respective null hypothesis. As for distractions and errors counted being different between having normal feedback and gradual feedback, but the evaluation leads to believe that the thesis cannot accept any hypothesis stating that normal visual feedback resulted in less distractions or errors and vice versa, since there is no statistic difference found while using ANOVA.

Hypothesis (H2): *Getting normal visual feedback results in objects being selected faster by users than having gradual display of feedback on the screen.*

Null Hypothesis (H2): *Getting normal visual feedback does not result in objects being selected faster by users than having gradual display of feedback on the screen.*

When talking about speed of selection differences we also cannot say with a statistically significant security that having visual feedback resulted in objects being selected faster than not having visual feedback. In terms of having a statistical difference between no visual feedback and gradual regarding selection speed, the comparison of averages with a histogram graph would lead to believe that normal visual feedback results in faster selection for the 28 participants' case, albeit the fact that ANOVA tests did not give statistical significance that there is a difference of selection speed between normal and gradual visual feedback. Therefore we also cannot accept the hypothesis 2 and have to reject it for the null hypothesis.

Hypothesis (H3): *Interactions with the same objects deliver a faster selection time than having to select objects closer to the respective partners.*

Null Hypothesis (H3): *Interactions with the same objects do not deliver a faster selection time than having to select objects closer to the respective partners.*

Hypothesis (H4): *Interactions with respective objects, i.e. the objects closer to users result in a faster selection time than selecting objects closer to the partners.*

Null Hypothesis (H4): *Interactions with respective objects, i.e. the objects closer to users do not result in a faster selection time than selecting objects closer to the partners.*

Now if we take into consideration the hypothesis formulated for interactions with the same object resulting in a faster selection time than respective or opposite objects, the mean calculations and comparison shows that interactions with the respective object, i.e. the object closest to the user seem to result in faster selection time than same or opposite, but statistically saying we cannot accept the hypothesis 3 either as the significance values were not low enough to be below 0.05. Therefore we have to reject hypothesis 4 as well. The null hypotheses though are relevant instead.

Hypothesis (H5): *Interactions with the same objects result in less distractions or errors than having to select objects closer to the respective partners or users.*

Null Hypothesis (H5): *Interactions with the same objects do not result in less distractions or errors than having to select objects closer to the respective partners or users.*

Hypothesis (H6): *Interactions with respective objects, i.e. the objects closer to users result in less distractions or errors than selecting objects closer to the partners.*

Null Hypothesis (H6): *Interactions with respective objects, i.e. the objects closer to users do not result in less distractions or errors than selecting objects closer to the partners.*

The interesting part of accepting or rejecting is about object interactions. Having to interact on the same object resulted in statistically significant less distractions than either respective or opposite objects, according to the ANOVA tests, therefore we can accept hypothesis 5 with regard to distraction count over the null hypothesis. As for error count we cannot do so statistically. In these cases the null hypothesis becomes relevant. Although the mean calculations suggested that interacting with the respective object would result in less errors than with the same or the opposite object. For hypothesis 6 we also cannot say with a statistical significance that interactions with the object closest to the user would result in less errors or distractions than interacting with the object closest to the partner.
We can say that distractions are logged subconsciously and are usually not noticed by the user themselves actively, also we pre-established a few rules for evaluation so that might affect the results as well. Errors however are logged by the users actively pushing a button to reset their selection which in hand results in errors being noted even for interactions resulting in no visual feedback where users do not see where the system thinks they are currently looking until the match has been made. Another point to mention is that error rate and speed of selection are affected by the system's limitation and the resulting user's impatience, as sometimes users resetted their selection when the system took too long to accept the right selection, which suggests problems with the system and tracking itself. To get to the ground of the problems it is necessary to reflect on the limitations that were observed during the user study.

8.2 Limitations of the System

During the conduct of the user study there were several limitations that we noticed that were not removed during debugging prior and some that were noticed because of faults in the hardware. First of all, because of the placement of the trackers gaze tracking sometimes did not collect data sufficiently for the outer parts of the display, i.e. the right area of the display for the left user and vice versa for the right user. This resulted in the furthest objects not being selected and thus correlation not being captured correctly.
Furthermore because of correlation being calculated every 500ms and gaze and object coordinates being added only if new gaze points arrive, if the memory usage results in lagging and latency, then there might not be enough points to actually calculate a sufficient correlation, which results in the wrong objects being chosen.
Another limitation of the system is the placement of the objects and the lines along which they move, i.e. sometimes when the movement of the right circle being linear overlaps with the movement of the left circle, the left circle was tracked and selected instead, as the system starts with checking correlation values with the left object first.
As for hardware limitations, one of the eye-trackers kept disconnecting at random times resulting in parts of the study having to be redone because the logging system did not finish logging but was interrupted. Furthermore sometimes the program would hang itself and stop working altogether, which can be a fault of eclipse of the program creating too much overload in terms of memory usage.
System limitations further included that the tracker did not recognise some users eyes which also resulted in one group having to reschedule their study time because the software needed to be re-installed.
If we try to classify the limitations we can come up with three groups of improvements for them: first of all the limitations involving correlation process, including the time, the sample size, the way coordinates are stored, etc. Secondly there are limitations concerning the way feedback is given to the users, which can be improved. These also include users not wishing for more effects or different colours, or only one type of feedback not sound and visual at the same time. The third limitation classification is about stimuli, i.e. the moving objects themselves.
To justify the limitations we can say that due to hardware having faults, such as loose contact or not tracking specific eye shapes or sizes or people wearing make up, etc. the collected gaze data can be limited. Also having less memory to use can result in latency or lagging of the movement which in its turn makes interaction strenuous for participants. Another point is due to the system not being debugged enough prior to conducting the study which in its turn result in problems arising during the study process. Furthermore the placing of the trackers was chosen as to not occlude the screen and obstruct the user's view of the stimuli, which in its turn result in problems while tracking gaze.
These limitations result in a few improvement points which encourage further works and research. This will be explained in the next chapter *Future Works*.

9 Future Works

Although we cannot significantly state that the system giving feedback to users necessarily resulted in faster or slower selection time and error rate, it was found mostly as very helpful by the students participating in the user study. With those qualitative opinions there are several components that still have to be further researched to improve the usability of the newly developed interface.
In this chapter we tried to classify the several possibilities open to further deepen research on visual feedback for multiple users interacting on the same screen based on the different properties: improving correlation calculations, upgrade stimuli movement, enhance feedback possibilities.

9.1 Improving Correlation

There are several ways to improve correlation and get a better correlation value. If we look at normally calculated correlation values they often take on a number between 0 and 1, with values closer to 1 resulting in an overall better correlation between two sets of data. Since the system calculates correlation values for both the x-coordinate and the y-coordinate, if we take the average between the calculated values it can often result in a lower overall correlation.
One step to increase the correlation can be to take higher dataset samples, i.e. to correlate more x- and y-coordinates between gaze movement and object position on the screen. To obtain a higher sample size we can either increase the time span between the actual calculations of correlation. For this particular system we opted to determine the correlation every 500ms, which can be increased to 1000ms or more. To obtain the right sample size to better distinguish which object was selected, i.e. which object movement correlated the most with the user's gaze pattern, it is necessary to test out multiple time spans in order to find the best fitting one, that does not take too long for the user themselves and relays the best correlation distinguishing.
Another way to improve the sample size added to the respective lists of object movement and gaze data would be to add gaze more frequently. The algorithm in our implementation adds gaze data and object movement data for every user only if gaze is detected and rightfully recognised as the user looking on the screen, thus meaning that collecting data samples does not happen at a fixed rate, in case the tracker does not provide the sufficient amount of gaze data.

9.2 Improving Stimuli

To enhance stimuli, there are a few possibilities that were mentioned by participants. Firstly, it might be advantageous to adjust the speed, so that it differs between the objects displayed on the screen. Right now the object movement speed we set is still rather similar, which might also affect correlation calculations. Furthermore the timing of trajectory movement could be adjusted as well, as to avoid false selections made by the system, if two objects find themselves in the same general area at multiple points during the interaction.
One participant in particular mentioned the use of other movement trajectories as well, i.e. having the objects move along the circumference of an arc or a triangle shape to try out whether adding different movement shapes would affect the overall selection process.

9.3 Improving Feedback

There are several ways to improve the overall given feedback. First of all the colours assigned to users should be contrasting from the background and second of all not be too bright for users, therefore it is necessary to test out which colours are the most *comfortable* to use in this case. Another point is that when having gradual feedback, the selection process should be sped up to avoid annoyance and or boredom coming up while having to wait for selection to be accepted by the system. In this case we need to find the right value as to still allow users time to interactively reset their selection before it is locket, but not having them wait too long. Another way to improve feedback is in optimizing the design of the interface. There are several ways to position the feedback, either inside of the stimuli or keeping it around the stimuli, but since this is a question of personal preference, this point is to be considered not as important as the prior.

9.4 Next Step and Future Works

As this research topic is in particular a test for usability of the system, it therefore heavily involves working with users and thus user studies. The logical step would hence be to do another study with the improvements included to test whether improving the stimuli movement would result in less participants complaining about strained eyes, as well as a faster selection of expected objects and less falsely selected other objects.
Another spin-off could be to research if distractions affect users, i.e. create an interface where multiple participants have to interact in a competitive manner and see if visual feedback results in participants taking more time to end their tasks.
If the tracking process is optimised and correlation calculations between gaze and other limitations are overcome for the desktop setting use of returning visual feedback for multiple users, it would be beneficial to see if an in the wild study would show that users appreciated visual feedback in terms of selection speed, error rate and distraction count. This would be the follow-up study of the one we conducted in this thesis, where we would display the improvements we made on our system until then.
Finally we should bring the study onto a public display to observe whether the placement of the eye-tracking device would result in different accuracy levels when tracking gaze and thus giving feedback would have to be adjusted accordingly.

10 Conclusion

Enabling gaze interaction for multiple user on public displays comes with challenges that concern not only accuracy of tracking devices and their placing according to the screen and user, but also in terms of how to give users in a collaborative environment feedback on which objects or spaces on the screen their respective partner is interacting with at the moment. Further challenges include the aspects of public displays in general: Most often they are larger-sized displays, usually mounted to a wall in a crowded and busy space, which on the one hand does not allow for sensible data to be displayed. On the other hand public displays are limited in the reachability for the user and the accessibility overall. Also, interactions on such screens are very susceptible to noise and distractions by foreign and external factors.
Gaze interaction was introduced on public displays to solve existent problems concerning the use of larger-sized screens, albeit discovering more challenges for research, such as the question how to allow for instant use in public spaces without having to calibrate tracking-devices.
This thesis aimed to provide several methods to show visual feedback for users interacting with public screens using a gaze-only approach. In this case the interaction method utilised gaze pursuit for users to actively select moving objects on the screen. Feedback was given in three separate ways, each however, including sound as indicator for correctly chosen objects: No, normal and gradual visual feedback. The user study conducted in regard to showing users several types of feedback and having them evaluate the types, while simultaneously logging speed of selection, error rate and distraction count during selection process, showed the following results: First of all, for speed averages and error averages we cannot statistically make the assumption that the reception of visual feedback as opposed to not getting feedback at all resulted in a faster overall selection time. In this case the conducted ANOVA tests did not show a statistical significant difference. The same phenomena is observed for error rates in terms of having visual feedback as opposed to not receiving feedback. For interaction divided in areas, such as interactions with the same, opposite and respective object, no significant differences were notices in terms of selection speed and error rate as well. As for distraction count however, we can statistically significantly say, that having to select the same object instead of opposite and respective object resulted in less distractions by the partner being recorded, as having to selection objects that were closer to oneself or closer to the partner. Furthermore we can also significantly assume that receiving no visual feedback resulted in less distractions than receiving visual feedback. However, in terms of the comparison between opposite object and respective object, as well as normal and gradual visual feedback, no significant differences in distraction count numbers were noted.
The results of the study evaluation showed that the system developed during the course of this thesis still needs improvement in terms of software specifications. Mostly the accuracy of the tracking devices for gaze were criticised by participants, which is understandable as the trackers were not positioned directly in front and middle of the screen, which resulted in contortions of gaze data recorded at the outer parts of the screen. Furthermore stimuli and feedback itself have to be adjusted to guarantee better usability.
The study albeit showing that having no feedback did not result in significantly faster or slower selections, less or more errors, illustrated a tendency of normal visual feedback resulting in faster selections as opposed to no or gradual feedback. Furthermore having to select the respective object, i.e. the object closest to the user supposedly resulted in faster selections for the particular group of participants. As for error rate it can not be said as clearly as for speed.
For future work it is important to ensure that the tracker is positions relatively centered to the display, if mounted, to ensure that the whole screen is seized accurately. Furthermore stimuli and feedback methods should be adjusted to occur faster for gradual feedback to keep the user's attention high during interactions on a public screen, as they mostly take place during the course of seconds. In addition to that it would be interesting to see whether incorporating the methods of feedback visualisations would give incentive and be helpful for more collaborative and teamwork

interactions on publicly placed displays, although we would need to also take into consideration which type of teamwork the displays would allow as confidential data should not be displayed on publicly accessible screens. Still, we find that having gaze pursuit interaction allows for spontaneous usage of public displays and having visual feedback would benefit the overall collaborative aspect that is important in everyday life and work. Still, we agree that in order to fully enable this concept on public display settings, more research needs to be done. As for now tracking works best using eye-tracking specific devices such as eye-trackers or glasses for wearable tracking. To use glasses for tracking on public displays in crowded areas seem to be less profitable, as they are small devices which can easily break or disappear, whereas having to use trackers for multiple users can become expensive and a challenge in terms of placing the users in front of the screen, as the trackers need to capture the user's eyes in order to map gaze correctly, as well as to be positioned according to the display to avoid peripheral contortions. The logical and challenging next step would be to enable gaze tracking fully for RGB camera use, as these can be integrated into the display without the need for user positions of the user having to position the tracker directly in regard to the user's eye position, but can according to Wood et al. track features regardless of resolution as long as the eyes are visible [68]. This is the goal to achieve in the future.

Contents of attached CD

- eclipse project with Tobii REX tracker use for feedback for multiple users (Java)
- Visual Studio project with CLM for tracking gaze attempts (RGB Camera Tracking)
- SPSS files and web reports of results for speed, errors, distractions
- exported Google Sheets containing data from study
- tobiisdk4j wrapper contents (not available online anymore)
- group task lists
- log files of user study

References

[1] ANOVA with Repeated Measures using SPSS Statistics. https://statistics.laerd.com/spss-tutorials/one-way-anova-repeated-measures-using-spss-statistics-2.php. [Online; accessed 26-May-2017].

[2] Box Plot: Display of Distribution. http://www.physics.csbsju.edu/stats/box2.html. [Online; accessed 25-May-2017].

[3] CLM-framework (a.k.a Cambridge Face Tracker). https://github.com/TadasBaltrusaitis/CLM-framework. [Online; accessed 27-May-2017].

[4] Commons Math - The Apache Commons Mathematics Library. http://commons.apache.org/proper/commons-math/. [Online; accessed 24-May-2017].

[5] CSE/EE486 Computer Vision I. Introduction to Computer Vision. http://www.cse.psu.edu/ rtc12/CSE486/. [Online; accessed 25-May-2017].

[6] EyeSee. http://eyesee-research.com. [Online; accessed 24-May-2017].

[7] EyeTribe Tracking Tool. http://www.theeyetribe.com/. [Online; accessed 27-May-2017].

[8] Minim. http://code.compartmental.net/tools/minim/. [Online; accessed 27-May-2017].

[9] OpenFace: an open source facial behavior analysis toolkit. https://github.com/TadasBaltrusaitis/OpenFace. [Online; accessed 27-May-2017].

[10] Processing Software. https://processing.org. [Online; accessed 27-May-2017].

[11] Tobii Gaze Tracking Devices. https://tobiigaming.com/products/. [Online; accessed 27-May-2017].

[12] xLabs - Eye and Head Tracking via Webcam. https://xlabsgaze.com. [Online; accessed 24-May-2017].

[13] AGOSTINI, A., MICHELIS, G. D., DIVITINI, M., GRASSO, M., AND SNOWDON, D. Design and deployment of community systems: reflections on the Campiello experience. *Interacting with Computers 14*, 6 (2002), 689 – 712.

[14] ALT, F., MEMAROVIC, N., GREIS, M., AND HENZE, N. UniDisplay x2014; A research prototype to investigate expectations towards public display applications. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)* (March 2014), pp. 519–524.

[15] ALT, F., SHIRAZI, A. S., KUBITZA, T., AND SCHMIDT, A. Interaction Techniques for Creating and Exchanging Content with Public Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), CHI '13, ACM, pp. 1709–1718.

[16] BALL, R., AND NORTH, C. Effects of Tiled High-resolution Display on Basic Visualization and Navigation Tasks. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2005), CHI EA '05, ACM, pp. 1196–1199.

[17] BALTRUSAITIS, T., ROBINSON, P., AND MORENCY, L. P. Constrained Local Neural Fields for Robust Facial Landmark Detection in the Wild. In *2013 IEEE International Conference on Computer Vision Workshops* (Dec 2013), pp. 354–361.

[18] BALTRUŠAITIS, T., ROBINSON, P., AND MORENCY, L. P. OpenFace: An open source facial behavior analysis toolkit. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (March 2016), pp. 1–10.

[19] BARNES, G. R. Rapid learning of pursuit target motion trajectories revealed by responses to randomized transient sinusoids. *Journal of Eye Movement Research 5*, 3 (2012).

[20] BEUN, R.-J., AND VAN EIJK, R. M. Conceptual Discrepancies and Feedback in Human-computer Interaction. In *Proceedings of the Conference on Dutch Directions in HCI* (New York, NY, USA, 2004), Dutch HCI '04, ACM, pp. 13–.

[21] BIEDERT, R. Tobii Gaze SDK (Java Wrappers). https://libraries.io/github/ralfbiedert/tobiisdk4j. [Online; accessed 25-May-2017].

[22] BRIGNULL, H., BRIGNULL, H., AND ROGERS, Y. Enticing people to interact with large public displays in public spaces. *IN PROCEEDINGS OF THE IFIP INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION (INTERACT 2003 2003* (2003), 17–24.

[23] BRIGNULL, H., IZADI, S., FITZPATRICK, G., ROGERS, Y., AND RODDEN, T. The Introduction of a Shared Interactive Surface into a Communal Space. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work* (New York, NY, USA, 2004), CSCW '04, ACM, pp. 49–58.

[24] BUXTON, W., FITZMAURICE, G., BALAKRISHNAN, R., AND KURTENBACH, G. Large displays in automotive design. *IEEE Computer Graphics and Applications 20*, 4 (Jul 2000), 68–75.

[25] CELEBI, F. M., KIM, E. S., WANG, Q., WALL, C. A., AND SHIC, F. A smooth pursuit calibration technique. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (New York, NY, USA, 2014), ETRA '14, ACM, pp. 377–378.

[26] CHATTERJEE, I., XIAO, R., AND HARRISON, C. Gaze+Gesture: Expressive, Precise and Targeted Free-Space Interactions. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (New York, NY, USA, 2015), ICMI '15, ACM, pp. 131–138.

[27] CONNOLLY, K. J. *Psychobiology of the Hand.* Cambridge University Press, 1998.

[28] CZERWINSKI, M., ROBERTSON, G., MEYERS, B., SMITH, G., ROBBINS, D., AND TAN, D. Large Display Research Overview. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2006), CHI EA '06, ACM, pp. 69–74.

[29] CZERWINSKI, M., SMITH, G., REGAN, T., MEYERS, B., ROBERTSON, G., AND STARKWEATHER, G. Toward Characterizing the Productivity Benefits of Very Large Displays. In *(2003) Interact 2003* (January 2003), IOS Press.

[30] DAVIES, N., CLINCH, S., AND ALT, F. *Pervasive Displays:Understanding the Future of Digital Signage*. Morgan amp; Claypool, 2014.

[31] FEIT, A. M., WILLIAMS, S., TOLEDO, A., PARADISO, A., KULKARNI, H., KANE, S., AND MORRIS, M. R. Toward Everyday Gaze Input: Accuracy and Precision of Eye Tracking and Implications for Design. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, ACM, pp. 1118–1130.

[32] HUANG, E. M., AND MYNATT, E. D. Semi-public Displays for Small, Co-located Groups. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2003), CHI '03, ACM, pp. 49–56.

[33] HUANG, E. M., MYNATT, E. D., AND TRIMBLE, J. P. When Design Just Isn'T Enough: The Unanticipated Challenges of the Real World for Large Collaborative Displays. *Personal Ubiquitous Comput. 11*, 7 (Oct. 2007), 537–547.

[34] HUANG, M. X., KWOK, T. C., NGAI, G., CHAN, S. C., AND LEONG, H. V. Building a Personalized, Auto-Calibrating Eye Tracker from User Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2016), CHI '16, ACM, pp. 5169–5179.

[35] JACOB, R. J. K. What You Look at is What You Get: Eye Movement-based Interaction Techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1990), CHI '90, ACM, pp. 11–18.

[36] JAKOBSEN, M. R., AND HORNB, K. Up Close and Personal: Collaborative Work on a High-resolution Multitouch Wall Display. *ACM Trans. Comput.-Hum. Interact. 21*, 2 (Feb. 2014), 11:1–11:34.

[37] KANGAS, J., RANTALA, J., MAJARANTA, P., ISOKOSKI, P., AND RAISAMO, R. Haptic Feedback to Gaze Events. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (New York, NY, USA, 2014), ETRA '14, ACM, pp. 11–18.

[38] KANGAS, J., ŠPAKOV, O., ISOKOSKI, P., AKKIL, D., RANTALA, J., AND RAISAMO, R. Feedback for Smooth Pursuit Gaze Tracking Based Control. In *Proceedings of the 7th Augmented Human International Conference 2016* (New York, NY, USA, 2016), AH '16, ACM, pp. 6:1–6:8.

[39] KHAMIS, M., ALT, F., AND BULLING, A. 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* (New York, NY, USA, 2015), UbiComp/ISWC'15 Adjunct, ACM, pp. 863–872.

[40] KHAMIS, M., BULLING, A., AND ALT, F. Tackling Challenges of Interactive Public Displays Using Gaze. 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* (New York, NY, USA, 2015), UbiComp/ISWC'15 Adjunct, ACM, pp. 763–766.

[41] KHAMIS, M., SALTUK, O., HANG, A., STOLZ, K., BULLING, A., AND ALT, F. 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* (New York, NY, USA, 2016), UbiComp '16, ACM, pp. 274–285.

[42] KHAMIS, M., TROTTER, L., TESSMANN, M., DANNHART, C., BULLING, A., AND ALT, F. EyeVote in the Wild: Do Users Bother Correcting System Errors on Public Displays? In *Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia* (New York, NY, USA, 2016), MUM '16, ACM, pp. 57–62.

[43] MAJARANTA, P., AND BULLING, A. *Eye Tracking and Eye-Based Human–Computer Interaction*. Springer London, London, 2014, pp. 39–65.

[44] MÄKELÄ, V., SHARMA, S., HAKULINEN, J., HEIMONEN, T., AND TURUNEN, M. Challenges in Public Display Deployments: A Taxonomy of External Factors. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, ACM, pp. 3426–3475.

[45] MÜLLER, J., ALT, F., MICHELIS, D., AND SCHMIDT, A. Requirements and Design Space for Interactive Public Displays. In *Proceedings of the 18th ACM International Conference on Multimedia* (New York, NY, USA, 2010), MM '10, ACM, pp. 1285–1294.

[46] PAVLOVIC, V. I., SHARMA, R., AND HUANG, T. S. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 7 (Jul 1997), 677–695.

[47] PELTONEN, P., KURVINEN, E., SALOVAARA, A., JACUCCI, G., ILMONEN, T., EVANS, J., OULASVIRTA, A., AND SAARIKKO, P. It's Mine, Don'T Touch!: Interactions at a Large Multi-touch Display in a City Centre. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), CHI '08, ACM, pp. 1285–1294.

[48] PÉREZ-QUIÑONES, M. A., AND SIBERT, J. L. A Collaborative Model of Feedback in Human-computer Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1996), CHI '96, ACM, pp. 316–323.

[49] PERRY, M., BECKETT, S., O'HARA, K., AND SUBRAMANIAN, S. WaveWindow: Public, Performative Gestural Interaction. In *ACM International Conference on Interactive Tabletops and Surfaces* (New York, NY, USA, 2010), ITS '10, ACM, pp. 109–112.

[50] PFEUFFER, K., ALEXANDER, J., CHONG, M. K., AND GELLERSEN, H. Gaze-touch: Combining Gaze with Multi-touch for Interaction on the Same Surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2014), UIST '14, ACM, pp. 509–518.

[51] PFEUFFER, K., ALEXANDER, J., CHONG, M. K., ZHANG, Y., AND GELLERSEN, H. Gaze-Shifting: Direct-Indirect Input with Pen and Touch Modulated by Gaze. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology* (New York, NY, USA, 2015), UIST '15, ACM, pp. 373–383.

[52] PFEUFFER, K., ALEXANDER, J., AND GELLERSEN, H. GazeArchers: Playing with Individual and Shared Attention in a Two-player Look&#38;Shoot Tabletop Game. In *Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia* (New York, NY, USA, 2016), MUM '16, ACM, pp. 213–216.

[53] PFEUFFER, K., VIDAL, M., TURNER, J., BULLING, A., AND GELLERSEN, H. Pursuit Calibration: Making Gaze Calibration Less Tedious and More Flexible. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2013), UIST '13, ACM, pp. 261–270.

[54] PFEUFFER, K., ZHANG, Y., AND GELLERSEN, H. A Collaborative Gaze Aware Information Display. 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* (New York, NY, USA, 2015), UbiComp/ISWC'15 Adjunct, ACM, pp. 389–391.

[55] RUSSELL, D. M., DREWS, C., AND SUE, A. *Social Aspects of Using Large Public Interactive Displays for Collaboration.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 229–236.

[56] SCHENK, S., DREISER, M., RIGOLL, G., AND DORR, M. GazeEverywhere: Enabling Gaze-only User Interaction on an Unmodified Desktop PC in Everyday Scenarios. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, ACM, pp. 3034–3044.

[57] SHIMIZU, J., LEE, J., DHULIAWALA, M., BULLING, A., STARNER, T., WOO, W., AND KUNZE, K. Solar System: Smooth Pursuit Interactions Using EOG Glasses. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct* (New York, NY, USA, 2016), UbiComp '16, ACM, pp. 369–372.

[58] STELLMACH, S., AND DACHSELT, R. Look &#38; Touch: Gaze-supported Target Acquisition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, ACM, pp. 2981–2990.

[59] STELLMACH, S., AND DACHSELT, R. Still Looking: Investigating Seamless Gaze-supported Selection, Positioning, and Manipulation of Distant Targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), CHI '13, ACM, pp. 285–294.

[60] TAN, D. S., GERGLE, D., SCUPELLI, P., AND PAUSCH, R. Physically Large Displays Improve Performance on Spatial Tasks. *ACM Trans. Comput.-Hum. Interact. 13*, 1 (Mar. 2006), 71–99.

[61] TURNER, J., ALEXANDER, J., BULLING, A., AND GELLERSEN, H. Gaze+RST: Integrating Gaze and Multitouch for Remote Rotate-Scale-Translate Tasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, 2015), CHI '15, ACM, pp. 4179–4188.

[62] TYERS, AND RAPPSILBER. BoxPlotR: a web-tool for generation of box plots. http://boxplot.bio.ed.ac.uk. [Online; accessed 25-May-2017].

[63] VELLOSO, E., WIRTH, M., WEICHEL, C., ESTEVES, A., AND GELLERSEN, H. AmbiGaze: Direct Control of Ambient Devices by Gaze. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (New York, NY, USA, 2016), DIS '16, ACM, pp. 812–817.

[64] VIDAL, M., BULLING, A., AND GELLERSEN, H. Detection of Smooth Pursuits Using Eye Movement Shape Features. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (New York, NY, USA, 2012), ETRA '12, ACM, pp. 177–180.

[65] VIDAL, M., BULLING, A., AND GELLERSEN, H. 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* (New York, NY, USA, 2013), UbiComp '13, ACM, pp. 439–448.

[66] WARE, C., AND MIKAELIAN, H. H. An Evaluation of an Eye Tracker As a Device for Computer Input2. *SIGCHI Bull. 18*, 4 (May 1986), 183–188.

[67] WILSON, F. R. *The Hand: How Its Use Shapes the Brain, Language, and Human Culture*. Vintage; 1st Vintage Books Ed edition, 1999.

[68] WOOD, E., BALTRUAITIS, T., ZHANG, X., SUGANO, Y., ROBINSON, P., AND BULLING, A. Rendering of Eyes for Eye-Shape Registration and Gaze Estimation. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec 2015), pp. 3756–3764.

[69] ZHAI, Y., ZHAO, G., ALATALO, T., HEIKKILÄ, J., OJALA, T., AND HUANG, X. Gesture Interaction for Wall-sized Touchscreen Display. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication* (New York, NY, USA, 2013), UbiComp '13 Adjunct, ACM, pp. 175–178.

[70] Zhang, Y., Müller, J., Chong, M. K., Bulling, A., and Gellersen, H. Gaze-Horizon: Enabling Passers-by to Interact with Public Displays by Gaze. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (New York, NY, USA, 2014), UbiComp '14, ACM, pp. 559–563.

[71] Zhang, Y., Pfeuffer, K., Chong, M. K., Alexander, J., Bulling, A., and Gellersen, H. Look Together: Using Gaze for Assisting Co-located Collaborative Search. *Personal Ubiquitous Comput. 21*, 1 (Feb. 2017), 173–186.