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

Master's Thesis

Gaze-based Interaction using Pursuits in VR environments

Carl Oechsner
oechsner.carl@gmail.com

Bearbeitungszeitraum: 1. 11. 2016 bis 7. 7. 2017
Betreuer: Mohamed Khamis
Verantw. Hochschullehrer: Prof. Dr. Florian Alt

Zusammenfassung

Der Einsatz von Eye-Tracking in virtueller Realität ist ein vielversprechender Ansatz, der neue Interaktions-Paradigmen und technische Verbesserungen ermöglicht. Im öffentlichen Raum, wo Virtual Reality zunehmend zum Einsatz kommt, zeigt sich weiteres Potential dieser Interaktiontechnik. Bisherige Studien zu öffentlichen Displays haben gezeigt, dass Nutzer eher zu Interaktion bereit sind, wenn diese sofort und ohne Vorkenntnisse möglich ist. Eye-tracking macht dies möglich, im Speziellen durch den Einsatz von Pursuits. Diese Technik hat bereits in einigen Studien bewiesen, dass sie die blickbasierte Auswahl von Objekten, die sich auf Displays bewegen, ohne vorherige Kalibrierung erfolgreich ermöglicht. Jedoch gibt es wenig Literatur, die sich mit der Implementierung von Pursuits in der virtuellen Realität befasst. Im Zuge der vorliegenden Arbeit wurde ein System entwickelt, um die Pursuits-basierte Interaktion mit Objekten in einer virtuellen Umgebung zu ermöglichen. Zur Evaluation der Leistungsfähigkeit und Anwendbarkeit wurde eine Benutzerstudie durchgeführt, woraus Design-Implikationen für Stimuli auf kreisförmigen Trajektorien abgeleitet werden können. Unterschiedliche Merkmale der Faktoren Größe, Trajektorienradius und Objektentfernung wurden auf ihren Einfluss auf Selektionszeit und Fehlerrate untersucht. Zusätzlich wurde unterschieden, ob der Nutzer sitzt oder durch die virtuelle Szene navigieren muss. Die Ergebnisse zeigen, dass das Gehen korrekte Selektionen erschwert und die Selektionszeit erhöht. Eine Erhöhung des Trajektorienradius kann dem entgegenwirken. Darüber hinaus wurden für das implementierte System zwei Anwendungsszenarien erstellt und qualitativ ausgewertet.

Abstract

Eye-tracking is a promising approach to enable new interaction paradigms and technical improvements for virtual reality. Furthermore, virtual reality is increasingly present in public settings, where interaction has to work instantaneously without any prior knowledge by the user or lengthy configurations. In several studies, Pursuits has proven to successfully enable gaze based selection of objects moving on displays without prior calibration, but little work exists on its implementation in virtual reality, which is the focus of this thesis. A system was developed to enable Pursuits interaction with objects in a virtual environment, and a user study was conducted to examine its performance and usability and to derive design implications for stimuli moving on a circular trajectory. For the factors size, trajectory radius and object distance different characteristics were investigated for their influence on selection time and error rate. Performance was tested having the user sit still and engage in a navigation task respectively. Results show that walking significantly decreases selection correctness and increases selection time. Increasing the trajectory radius, however, can counteract these effects. Additionally, two use case scenarios were created for the implemented system and evaluated qualitatively.

Task

Gaze-based interaction using Pursuits is increasingly becoming popular for gaze-enabled systems. This is mainly due to it being natural to use, robust to the Midas touch problem, and calibration-free. While there are VR systems that employ eye tracking already, these systems are limited to interaction via dwell time. In this project, the aim is to explore the use of smooth pursuit eye movements for gaze interaction in virtual reality.

Tasks

- Reviewing state of the art in eye tracking in virtual reality
- Implementing the Pursuits algorithm in VR
- Exploring the design space of Pursuits in VR
- Conducting a user study to compare different configurations for on-screen stimuli
- Implementing and evaluating realistic scenarios that showcase the use of Pursuits in VR
- Analyzing and reporting the results

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

München, 7. Juli 2017

. . . . . .

Inhaltsverzeichnis

1 Introduction

Virtual reality (VR) technology has experienced an enormous hype over the last two years, which started with the launch of the Oculus Rift in and HTC Vive in early 2016. VR is described as the "next big thing" that "will shape our everyday life like the smartphone" [1] [2]. Since then, more and more applications for VR have been developed and by May 2017 about 3,4 million VR games have been sold via *Steam*, a major online distributor for game content [3]. The elaborate tracking and high resolution screens of current headsets provide users with a natural feeling of actually being in the scene. There is almost no perceptible lag that could lead to motion sickness. Applications for VR reach from first-person shooters [4], over room-planning[1] and 3D design and visualization software[2], multi-user environments, in which people can work, learn[5] or play games[3], to stroke recovery therapy [6].

VR is also increasingly employed in public settings. Audi, as one of the first car manufacturer, has introduced virtual reality headsets in their flagship stores that allow potential buyers to virtually sit in their future car and customize it [7]. This helps provide a realistic, immersive experience without being restricted to the models in stock. Another area, in which VR enjoys increasing popularity is museums. Visitors are able to discover exhibits that are normally behind glass or prehistoric environments that are long gone [8] [9].

As rendering imagery for virtual scenes is pretty GPU-intensive, powerful graphics adapters are needed to provide a lagging-free, detailed experience. Together with the headset itself, bringing virtual reality into the living room becomes pretty costly. A new technique is currently investigated to reduce image quality in areas, the user is not actually looking at, and increase it at the position where the focus is, which can significantly reduce rendering cost [10]. This requires eye-trackers to be built into the VR headset. Appropriate extensions for Oculus and Vive headsets are already available[4] and in fall 2016, the first VR headset with built-in eye-trackers was launched[5].

There are more benefits of integrating eye-tracking into virtual reality. Gaze is a natural, intuitive way of interacting with our environment and indicates where our attention is focused at the moment or what we're going to do next. Furthermore, it is part of the non-verbal communication between humans and regulates interaction [11]. Especially in collaborative tasks, gaze information can improve communication by supporting visual grounding even more than verbal information [12] [13]. Therefore, eye-tracking as an input modality for virtual reality can achieve new and more seamless ways of interaction. It can allow "real" and artificial characters in a VR application to react to where the user's attention is directed or even to emotions [14]. Most virtual reality headsets come with controllers as the only means of interacting with objects in the virtual setting. Using gaze as an additional input can enable multimodal interaction: triggering an action or selecting an object is possible while the user's hands are occupied.

Although modern eye trackers and especially head-mounted eye-trackers provide a considerable accuracy, most implementations rely on absolute gaze data. To obtain this data, a calibration process is required that maps data from the trackers to screen coordinates [15]. The calibration process usually involves a procedure of successively looking at several targets that has to be repeated every time the headset is put on, as it is never exactly in the same position as the previous time. This small deviation can affect accuracy of the gaze data due to the short distance between camera and eyes, so a re-calibration is required. Calibrating is time-consuming and hinders a spontaneous use of the technique. Especially, when used in a public setting, it must be possi-

[1] Room Designer VR at Steam - `http://store.steampowered.com/app/601620/Room_Designer_VR/` (visited 2017-07-04)

[2] Tilt Brush at Steam - `http://store.steampowered.com/app/327140/Tilt_Brush/` (visited 2017-07-04)

[3] Rec Room at Steam - `http://store.steampowered.com/app/471710/Rec_Room/` (visited 2017-07-04)

[4] Pupil Labs - Store - `https://pupil-labs.com/store/` (visited 2017-07-04)

[5] FOVE Eye Tracking Virtual Reality Headset - `https://www.getfove.com` (visited 2017-07-04)

ble to instantaneously interact with a system without prior knowledge or protracted adaptation procedures to prevent user frustration [16] [17].

Vidal et al. have developed a method named "Pursuits [that] does not require prior knowledge from the user, as it exploits the natural occurrence of pursuits when a moving object is being followed" [18] and allows for spontaneous, calibration-free gaze interaction. The technique detects fixations on moving objects by correlating eye movements with object movements and therefore removes the need for calibration. It works as long as both motions of eye and object resemble each other. By employing Pursuits approach in virtual reality, spontaneous interaction would be enabled without the need for prior configuration or additional input devices. Since there is little work on implementing Pursuits for virtual reality, this was decided to be the topic approach in this thesis.

This document is structured as follows: section 2 covers further information on eye-tracking and Pursuits and how it can improve interaction in virtual reality, which is followed by a description of the design space for the application of Pursuits in virtual reality. The development of a software which uses tracking data to employ the Pursuits algorithm is described in section 4. For its evaluation and the derivation of design guidelines, a trial study and a main study were conducted. Procedure and results of these studies are covered in section 5. The findings of this thesis are summarized in section 6 and finally, section 7 illuminates possible restrictions of the found results and proposes interesting questions for future studies.

2 Conceptual Framework

This section is aimed towards conveying more insight in the matter this thesis is based on. The first section "Eye Tracking in HCI" gives a survey of the development of eye tracking over time and the underlying concepts. Thereafter a short summary is given of fields in which eye-tracking is currently employed. Subsequently, the Pursuits technique and results from studies investigating it are described. Finally, the advantages of eye-tracking in virtual reality are addressed specifically and how Pursuits could improve their implementation.

2.1 Eye Tracking in HCI

Long before eye movements were used to interact with a system, they were mostly recorded and used for usability testing (e.g. eye movements while reading or of pilots using cockpit controls) or psychological research (e.g. to study cognitive processes). Until the first head mounted eye-tracker (HMD) was invented in 1948, even the smallest head movements or a person's pulse could have an impact on the recorded data [19]. Eye-tracking technology was constantly improved over the next decades by allowing more natural head movements and reducing the size of the whole system while improving accuracy. While in the 60s, the apparatus to track eye movements still occupied a considerable amount of the subject's field of view and one had to decide between accuracy and convenience [20], in the 70s Cornsweet and al. introduced a tracking device based on corneal reflections that was impervious to translations of the eye and achieved a tracking accuracy of 1 arc sec [21] [22]. By the 1980s subjects were already able to move within a space of one cubic foot and even glasses were tolerated [23]. The freedom of movement was achieved by using an external light source that illuminates the user's eye, which results in a glint on the eyeball that remains on the same position while the eye is moving. A light spectrum of near infrared was used as it does not affect the user's vision. By obtaining both the position of the glint and the position of the pupil, translations can be differentiated from rotations and therefore the user can move relatively freely. The first successful attempts were made with a video based eye-tracking system to use the line of sight as an input mechanism for " 'menu' selection [or] phone number look-up" [24] in an "office-like" environment. This non-intrusive technique is still the basis of today's video based eye-tracking systems (e.g. Tobii [6]) and known as pupil center corneal reflection.

This technique delivers very accurate results but one of its main drawbacks is the need for calibration. The reference points that pupil and corneal reflection constitute are not enough to determine which part of a screen the subject is looking at. The calibration procedure in which the user is usually looking at a number of points appearing on the screen is needed to map the glint-pupil vector to distinct screen coordinates [25]. Mapping accuracy decreases the more the user moves away from the original position in which the calibration took place. Many attempts have been made to reduce or abolish the need for calibration [26] [27] or to compensate mapping errors [28], as the calibration procedure "is a process that presents users with a task that is of poor usability, often experienced as difficult, and frequently described as tedious" [29]. All these drawbacks still decelerate the establishment of gaze as a dependable input mechanism, although eye-tracking technology is as affordable as never before.

Head mounted eye-trackers allow the user to move freely, as the cameras are attached to the users head and can track the eyes continuously from the same position. Employing head-mounted eye-trackers also allows for leaving the lab environment and conduct user studies under natural circumstances. However, this newly obtained freedom causes new problems: in real-world settings different light incident has an influence tracking accuracy and especially infrared-based trackers struggle with sunlight [30].

[6] How do Tobii Eye Trackers work? - https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/ (visited 2017-07-04)

Systems that use gaze as single input modality face the question of how to differentiate between the user exploring an object and intentionally selecting it, which constitutes the so-called Midas touch problem. If no distinction is made, simply every object is selected which the user looks at, hence the name. To cope with this problem, several methods exist: blink-, dwell- and gesture-based selection. The first uses a number of intentional blinks of the eye as the signal to perform a selection. The dwell-based method uses a delay until a fixated object is selected. The longer this dwell-time is, the more likely is the user's intention. The third relies on intentionally performed eye-movements. Either the user makes one or several eye movements that would not occur naturally or fixates on a defined area of the screen that triggers a certain action. The most common one is the dwell-time approach [11] [31]. However, fixating on a static object for a longer amount of time is unnatural, as our eyes typically fixate on a position no longer than 600 ms to 1500 ms [32] and the user must be careful not to investigate an object too long. Both, long dwell times and unintentional selections lead to user frustration the therefore reduce usability [33].

2.2 Areas of Application for Eye-Tracking

In his paper, Duchowski made a hierarchic approach to categorize the areas in which eye-tracking is currently employed [34]. At the top most layer, he distinguishes between diagnostic and interactive eye-tracking systems. In diagnostics, tracking data is used to determine how people process information as eye movement directly represents where a person's attention is directed. This insight can be used e.g. in psychological research to draw conclusions on cognitive processes and information processing [35] as well as in usability research [36] [37]. Tracking a person's eyes while evaluating an interface, software or procedure gives additional insights into the users' behavior, e.g. if they are able to proceed with purchasing an item in an online store interface. For instance, a higher number of fixations cannot only indicate an increased amount of interest on a particular part of a screen but can also be interpreted as uncertainty or a structure being overly complex [38]. This information is also used in marketing to increase effectiveness and even drive the viewers' attention in an advertisement [39].

The second group of eye-tracking systems utilizes gaze as a means of interacting with a system. Besides using eye-tracking as replacement of traditional input periphery, many attempts have been made to find new interaction paradigms that exhaust its possibilities. For example, gaze input is employed in research on interfaces enabling physically challenged users to interact with a computer or tablet [40] [32]. New forms of interacting with video games have been investigated that extend traditional controller input and allow for a more immersive gaming experience [41] [42] [43]. As with gaze control, there is no observable physical contact with a system; it can be used as a safe authentication method that prevents shoulder-surfing attacks [44] [45] [46]. Applications of gaze are also found in the field of teleoperation: This comprises techniques to operate in potentially dangerous or hazardous environments without the need for a human to be physically present, as drones or robots perform the required actions. Successful attempts have been made to steer these entities by using gaze based interfaces [47].

Considerable research has been carried out on enabling eye-tracking for public displays [48] [16]. Gaze based interaction is very suitable in this scenario as it does not require any additional input devices and users can access content even if the display is out of their physical reach. Particularly in this context, it is important for the system to be usable without any prior knowledge from the user, that it catches the user's attention and that no additional configuration, such as calibration, is needed.

2.3 Pursuits

When it comes to spontaneous, calibration-free eye-based interaction, Pursuits is the most promising contribution of the past years [18]. Its idea is simple and efficient: eye-movement data gets

compared with the movement of an object on an interface by performing a Pearson correlation between both the object's trajectory and the gaze data trajectory. If the resulting coefficient is high enough, the user is most probably looking at the respective object. Until this threshold is met, some time passes which makes Pursuits a dwell-based approach, but with a significant advantage: following moving targets is a natural process that occurs automatically, as opposed to fixating on the same position. Hence the dwell concept does not have to be explained to the user. In their paper, Vidal et al. analyzed different trajectories, object counts and object speeds and additionally give recommendations to implement such as correlation window size and correlation threshold.

How does this work? When a person wants to look at an object the eye needs to be positioned in a way that its image falls on an area on the retina called fovea, which is the region with the sharpest vision. To achieve this, the eye performs a saccade, a "sudden, ballistic and nearly instantaneous" [31] movement after which the eye is in the correct position, except for very large movements where a subsequent "correctional saccade" is possible. A saccade is usually followed by a fixation that lasts between 200 and 600 ms, if the target is stationary. If the eyes follow a moving object, a slower, continuous movement occurs called "smooth pursuit", which has the same velocity as the object being followed [31] [49].

The Pursuits approach is making use of this phenomenon, thus removing the need for a calibration phase because the process does not rely on tracking data being mapped exactly to screen coordinates and works as long as received tracking data resembles the object movement on the screen. Trajectories are recorded for both eye coordinates and object coordinates and associated using Pearson's product-moment correlation.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

Equation 1 describes the calculation of the product-moment coefficient for two trajectories X and Y: $\bar{a}$ is the mean of all samples in A, n is the sample size, which has to be the same in both X and Y. Furthermore, a threshold has to be defined, which has to be exceeded for an object to be considered as selected and a time window w, defining how long trajectory coordinates are kept in the trajectories. Vidal et al. investigated the effect of object parameters trajectory shape, speed and number as well as the correlation parameters w and Pearson threshold on the correctness of detection. They observed a drop in detection rate for an object count of eight or higher. A minimum difference of 200 px/s was determined to be sufficient for the algorithm to distinguish between two objects moving along the same linear trajectory[7]. Furthermore, a correlation threshold above 0.5 and a longer correlation window w is recommended to achieve a higher selection correctness. When all objects were moving at the same speed, there was no significant change in detection performance. They achieved a A circular trajectory allowed for robust detection of up to 15 objects whereas performance dropped for more than four objects on linear trajectories. The effect of speed was considered yet had no strong influence, if all objects move at the same speed. Although circular trajectories seemed to achieve the most robust detection, only one constellation with a distinct speed and radius has been tested.

Esteves et al. used circular trajectories to interact with smart watches using a head mounted eye-tracker [50]. They used the same correlation method for selecting control elements that were individually orbited by little dots, once on a Laptop screen with a stationary tracker and once on a smart watch with a head-mounted tracker. Following this dot with the eyes would result in the respective control to be selected. As for the trajectory of the dots, different diameters and angular speeds were tested for performance. They found that with a medium speed of $120°s^{-1}$

[7]For this specific setting a speed difference of 200px/s approximately corresponds to an angular speed of 3.04 degrees/sec if the object starts from the screen center. Visual angles decrease for the same on-screen distance the further it is from the screen center.

96% correct selections were possible on the laptop screen for single targets and 83% on the smart watch screen with up to eight targets, using a threshold of 0.8 and a window size of one second. On the laptop screen a larger radius had a positive effect on correct detections and on the watch it had no significant effect, unlike speed that led to better results, when slower. This difference in results shows that performance is dependent on the hardware setting and thus Pursuits implementation for virtual reality has to be re-evaluated. The object trajectories were displayed at a distance of 65 cm from the laptop screen and 35 cm from the watch. Objects in a virtual reality headset are few centimeters from the eyes but shown with a virtual depth, which can lead to very different performance results for Pursuits. Furthermore, the results from "Orbits" were gathered under lab conditions and no data was gathered under typical use cases like while walking. However, the robustness of circular trajectories has been confirmed.

2.4 Eye-based Interaction in VR

As mentioned earlier, there are eye-trackers being produced as extensions for existing virtual reality headsets. As they are attached to the device, pointing towards the user's eyes, they have the advantages of head-mounted eye-trackers (see section 2.2), while getting rid of one of their drawbacks: as they are positioned inside the headset where no light can enter, changing light conditions don't impede tracking accuracy. The rising interest in the combination of virtual reality and eye-tracking is further proven by the launch of *Fove*, which is the first headset with built-in eye tracking, which was launched in fall 2016 after it raised almost double the required amount on Kickstarter[8]. The possibilities of enabling gaze in games and other applications are manifold and have been investigated in gaming industry and research. As the gaze direction represents the actual area of interest of a person, this enables the design of reactive content that adapts to the user and constitutes an efficient interaction modality for virtual reality environments [51]. In their work, Tanriverdi and Jacob state that gaze based pointing is faster than pointing at an object with the hands in virtual environments, especially for remote objects [52], which is an advantage for instance in shooting games or other applications where aiming is crucial.

In collaborative virtual environments, like *Rec Room*[9], gaze can be used as means of non-verbal communication. Like in a real world setting, seeing another person's view helps improve situation awareness and conversational grounding [12] and can thereby help regulate interaction and turn-taking [11] [53]. Solutions are already being developed to integrate eye movement in avatar interaction [54].

Another aspiring technique that makes use of gaze data is called "Foveated Rendering", which reduces rendering load for the graphics adapter that the headset is connected to and that is currently being investigated [55] [56]. It uses the user's gaze to determine which area of the image is currently focused and reduces rendering quality in areas that are further away from the focus point.

All the instanced examples in this section are able to significantly improve user experience in virtual reality. However, with traditional eye-tracking they all are in need for calibrated gaze data, which suffers from the drawbacks described in section 2.1. Using Pursuits could increase their applicability, especially in public settings, where users must be able to interact with the system intuitively and without prior instruction or calibration.

Virtual reality is a whole new design space for utilizing Pursuits. As no literature has been found that has tested its applicability or performance in a virtual environment, basic research has been conducted in this thesis, which structure is orientated towards previous research: investigating different target and trajectory properties for their influence on selection time and precision. Therefore, a software has been created, which uses the correlation algorithm used by Vidal et al.,

[8] FOVE: The World's First Eye Tracking Virtual Reality Headset - Kickstarter - `https://www.kickstarter.com/projects/fove/fove-the-worlds-first-eye-tracking-virtual-reality?lang=de` (visited 2017-07-04)

[9] Rec Room at Steam - `http://store.steampowered.com/app/471710/Rec_Room/` (visited 2017-07-04)

to investigate these properties. The implementation of this software is described in section 4. In the following section, the aforementioned design space is specified.

3 Design Space

Pursuits in 2D relates on objects moving across the screen, while in most common settings the user is stationary and ideally resides in a “sweet spot” in front of the eye tracker, where eye tracking can be performed best. When transforming this idea to a three-dimensional space, content and user are no longer separated and interacting elements move in the same virtual environment. Furthermore, the user stays in the “sweet spot” at all times as eye trackers are attached within the headset in a fixed position relative to the user’s head, no matter if he or she is moving or not. As the user is practically another object in the virtual world, his or her position in relation to interactive objects are always known to the software. To enable interaction using Pursuits, one needs to understand and define which relations can exist between user and object. There were two states defined for user and object respectively and the combinations of these states constitute the relation in which interaction can emerge.

3.1 User Bound Objects

User bound objects perform translational motions in respect to the position and orientation of the user’s head. Thus, depending on the character of the motion, it appears to the person as if the object stays in the same region within the field of view. For example, an object could be configured to move on a circular trajectory in the user’s vision and looking at it would trigger a certain action. Regardless of the movement of the user, it will always be clear to them where to look for this action to be executed, which is comparable to a head-up display. In a stationary setting, this could be used for authentication purposes, such as for in-app purchases or to access private information. Other people, who might be in the same environment – real or virtual – would be unable to trace any physical movement as is the case if authentication is performed with a real controller. In case the user is moving, looking at objects placed in the peripheral field of view could select certain tools while the user continues to perform a main task with the controller, like pointing, writing or shooting. The correlation of movements of user bound objects with eye movements makes use of object coordinates that stem from the coordinate system of the user, and not the world. The x-axis of the user coordinate system is always situated horizontally in the field of view, so that an object movement to the right also leads to an eye movement to the right, independently of head tilt or rotation. Accordingly, following an object movement along the positive y-axis causes the eyes to move upwards.

3.2 World Bound Objects

As opposed to the user bound objects, world bound objects are anchored and move in respect to the coordinate system of the virtual scenery, i.e. they are not influenced by the way the user moves. To detect any fixations using Pursuits, the system does not only need to trace the object’s movement but also the user’s orientation towards the world coordinates. If the user’s eyes follow an object that is moving horizontally with a given amplitude in a way that the user’s line of sight is perpendicular to the object’s trajectory, the two movements can be correlated as they occur in the same velocity and orientation. Moving around the object causes its trajectory to become distorted from the user’s standpoint. Now, the horizontal movement as seen from the user is much slower just like the eye movements (see fig. 3.1). This exacerbates correlating the two movements and even inhibits it if the user’s line of sight is collinear with the object’s trajectory. Furthermore, if users follow an object by rotating the head instead of the eyes, eye pursuits are minimized. To take this into account, eye rotations have to be compensated for head tilt and combined with head rotation, before they can be correlated with object coordinates. This is possible, as the system is constantly aware of the exact position, tilt and rotation of the user’s head.

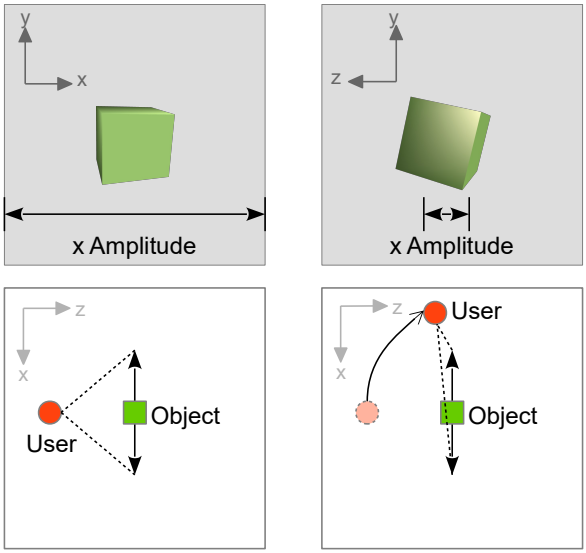


Figure 3.1: Depiction of the parallax effect occurring in a world-bound setting. The upper row represents the view from the user. The bottom row depicts the same scene from bird's eye perspective.

For this thesis, it was decided to focus on investigating the user-bound setting both for a stationary and a moving user. The user-bound setting resembles the settings, in which Pursuits have been implemented in previous research. If proven a working concept for virtual reality, the next step is to determine how Pursuits can be implemented for world bound settings. In the next section, the development of a system that detects fixations on user-bound objects using Pursuits is described.

4 Implementation

In this section, functionality of the implemented system is described in detail. In the first two subsections, the hardware setup is depicted, including how the distinct entities communicate with each other and how gaze data is acquired. The third subsection delineates the interpretation of this data and the specific implementation of the Pursuits approach, including descriptions of the central software components. A procedure to measure the field of view of a user wearing a virtual reality headset, is described in the last subsection of this chapter.

4.1 Virtual Reality Headset

An HTC Vive[10] was used to present stimuli to the users. It equipped with two OLED displays each with a resolution of 1080 x 1200 pixels and running at a refresh rate of 90 Hz. In the foremost setting the user has a field of view of up to 110°. Two small boxes called "Lighthouses" that are ideally positioned at two opposing corners of the intended "play area", constantly flood the room with a light pattern to enable the headset and controllers to determine their position. Inside a Lighthouse two lasers rotate at 60Hz, one horizontally, one vertically, and additionally 20 stationary LEDs flash at once at the same frequency, which serves as synchronization pulse. The headset (and sensors) are equipped with an array of photo sensors that calculate its position based on the time differences of when the light reaches them [57]. Additionally, the headset features a front camera enabling for Valve's *Chaperone* system that warns users if they come too close to an object in reality like walls, chairs and other furniture.

HTC cooperates with *Valve Corporation*[11], whose digital distribution platform *Steam*[12] is used to distribute virtual reality game content for the Vive headset, named "SteamVR". Valve collaborates with *Unity Technologies* and offers native support for SteamVR to develop virtual reality games with Unity 3D[13] [58]. The SteamVR-Plugin[14] allows for access to the virtual reality system including the position and orientation of headset and controllers, controller input and play area bounds. All this information is included in a prefab ("[CameraRig]"), which is included in this plugin and can simply be inserted into any scene in Unity, which can then be played and explored with the Vive headset. Merely the Main Camera object, which is a default object in every new scene, has to be removed. The prefab consists of an object for each controller and the user's head respectively, which are represented at the correct positions relative to the play area and can be included in the game logic.

4.2 Eye Tracking

To enable eye-tracking the HTC Vive was equipped with binocular eye-trackers by *Pupil Labs* with an accuracy of up to 0.6° of visual angle [59]. Each of these cameras is attached to a frame that can be snapped onto the lenses of the headset and that is fitted with infrared emitters[15]. The camera focus can be adjusted manually. The camera images are accessed and interpreted by *Pupil Capture*, a software that builds a virtual 3D model of the eye in the image and calculates and provides information like eyeball center, pupil center and gaze point (if calibrated). As the cameras are situated below the lenses and face towards the location of the eyes. It depends on the user's physique whether the eyes will be within camera range or not. The Vive headset allows to adjust the distance between lenses and eyes. With the shortest possible distance, the experience is the

[10] Vive | Discover Reality beyond Imagination - `https://www.vive.com/de/` (visited 2017-07-04)

[11] Valve Corporation - `http://www.valvesoftware.com` (visited 2017-06-09)

[12] Steam - `http://store.steampowered.com` (visited 2017-06-09)

[13] Unity 3D - `https://unity3d.com/de` (visited 2017-06-09)

[14] SteamVR Plugin by Valve Corporation - `https://www.assetstore.unity3d.com/en/#!/content/32647` (visited 2017-06-09)

[15] Pupil Labs - HTC Vive Binocular Add-On `https://pupil-labs.com/store/#vr-ar` (visited 2017-06-09)

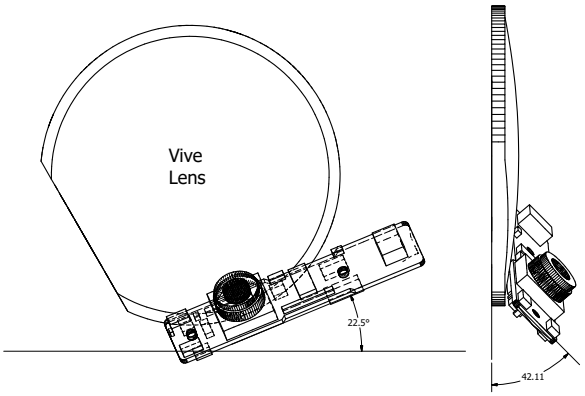


Figure 4.1: Drawing of a right lens of the HTC Vive with the Pupil Camera attached. The camera is rotated and tilted in a way that the lens can focus on the pupil of the user. For the left lens the camera is rotated by 180° so that the lens is at the right edge instead of the left. This also means that the two cameras have two different coordinate systems. (Picture was generated from a CAD file received from Pupil Labs)

most immersive but most of the time the eyes turned out to be at the upper limit of the camera image and the pupils were not visible when the user looked upwards. Increasing the distance, solved this problem and allows for capturing the whole eye but causes blurriness in the outer region of the field of view.

Any program on the same computer or the same network can receive data from Pupil Capture via TCP which is enabled by the *Pupil Remote* plugin, which is part of the Capture software. When this system was first developed, Capture was only available for Linux and so the trackers had to be connected to a Linux machine on the same network with an extra USB-C cord. Simultaneously, the VR headset itself had to be connected to a Windows machine that ran the Vive software. Communication with Pupil Remote is based on *ZeroMQ*[16]. ZeroMQ is an asynchronous networking library aimed for sending messages between distributed applications. It provides different patterns that can be implemented. The basic ones are Request-reply, Pub-Sub, Pipeline and Exclusive pair for each of which ZeroMQ provides the matching sockets.

The connection procedure passes off as follows: Pupil Capture uses both Request-reply and Pub-Sub (Publisher-Subscriber) patterns. Every time Capture is started it sends *Pupil* and *Gaze* data via a different port, which has to be determined first by sending a request. Thus, a *Request Socket* has to be created to connect to the respective address and port (standard for requests is 50020, but it can change). After the connection is established, a request frame with the message "SUB_PORT" has to be sent via this socket. As messages received by ZeroMQ ports are queued, the Request Socket has to be checked for a response, which is a message containing the port number on which Pupil Remote publishes data packages. Next a *Subscriber Socket* has to be created that connects to this port on the same address. This socket can now subscribe to a certain topic; “gaze” to receive gaze data for both eyes, “pupil.0” to receive raw pupil data for the right eye and/or “pupil.1” for the left eye.

Messages can now be read from the Subscriber Socket and consist of two frames: The first frame contains the topic of the message, the second part contains the message body, which is encoded using *MessagePack*[17], a binary serialization format like JSON but supposedly faster. Unpacking the message with MessagePack, results in a `MessagePackObject`, which can be converted to a simple string. This string is to be decoded using *JSON*[18] into a `PupilData3D`-Object which contains all the needed information for gaze based tasks (see Listing 4.1).

[16]ZeroMQ `http://zeromq.org` (visited 2017-06-09)
[17]MessagePack - `http://msgpack.org` (visited 2017-06-09)
[18]JSON - `http://www.json.org` (visited 2017-06-09)

Unity is a state of the art game developing software. Scripts can be attached to every object in the game that define behavior like movement, appearance, size or rotation. Therefore, it is very suitable not only to create games but to design testing environments for on-screen content like it was done in this thesis. The script language can be either Javascript, Boo or C#. The latter is the choice for this project. The required libraries described in the last paragraph are all available for C#: *NetMQ*[19] is a native ZeroMQ-port for C# and *msgpack-cli*, which is "an implementation of the MessagePack specification for the .NET CLI"[20]. A JSON library does not have to be imported, as it is part of of the Unity Engine (`UnityEngine.JsonUtility`).

Usually libraries in C# are bound by using the `using`-keyword at the beginning of the code, provided a pre-compiled .dll-file exists in the project folder. Current versions of NetMQ require that a certain version of the .NET Framework is installed on the machine on which features they rely on. In this case NetMQ relies on the *AsyncIO*[21] package for which .NET Framework v4.0 is required, although it uses none of the features of version 4.0.

Unity itself comes with *Mono*, an "open source development platform based on the .NET Framework"[22]. However, the .NET Framework it is based on, currently has version 2.0.5x, so the standard DLLs for NetMQ, MsgPack and AsyncIO will not work. The source projects for all three of them can be downloaded from GitHub and using e. g. Microsoft Visual Studio, custom DLLs can be compiled. It has to be made sure that before compilation, the target framework is set to ".NET Framework 3.5". Additionally, in the Unity Project settings, API Compatiblity level has to be set to ".NET 2.0".

```
public class PupilData3D
{
    public double diameter;
    public double confidence;
    public ProjectedSphere projected_sphere = new ProjectedSphere();
    public double theta;
    public int model_id;
    public double timestamp;
    public double model_confidence;
    public string method;
    public double phi;
    public Sphere sphere = new Sphere();
    public double diameter_3d;
    public double[] norm_pos = new double[] { 0, 0, 0 };
    public int id;
    public double model_birth_timestamp;
    public Circle3d circle_3d = new Circle3d();
    public Ellipse ellipese = new Ellipse();
}
```

Listing 4.1: : **The PupilData3D class**: Data received from Pupil Capture. `confidence`: states how sure the pupil detector is with the measurement, `id`: ID of the eye (0: right, 1: left), `norm_pos`: position in the eye image frame in normalized coordinates, `timestamp`: timestamp of the source image frame. Whole data description at `https://github.com/pupil-labs/pupil-docs/blob/master/user-docs/data-format.md`

[19] NetMQ - `https://netmq.readthedocs.io/en/latest/` (visited 2017-06-09)

[20] GitHub - MessagePack implementation for Common Language Infrastructure / msgpack.org[C#] `https://github.com/msgpack/msgpack-cli` (visited 09-06-2017)

[21] NuGet Gallery - AsyncIO - `https://www.nuget.org/packages/AsyncIO/0.1.26/`

[22] Mono | About Mono - `http://www.mono-project.com/docs/about-mono/` (visited 2017-06-09)

Fortunately, Pupil Labs provides a Unity project[23] that comes with custom DLLs for Unity and includes a class called `PupilGazeTracker`, which handles communication with the Capture software. It also provides an Event other classes can subscribe to that gets fired every time a `PupilData3D`-Object is received and its Confidence value is above a certain threshold. The contents of this object are listed in Listing 4.1.

4.3 Interpretation of Gaze Data and Implementing Pursuits

As mentioned before, Pupil Capture provides two kinds of data: raw *Pupil* data and calibrated *Gaze* data. Originally, raw data should have been used to pursue the completely calibration-free approach. In the Pupil raw data, one can find a variable called `circle_3d_normal` that represents the direction in which the pupil points and would identify the value of choice to correlate with. Unfortunately, this vector is oriented in the camera coordinate system, which has the same orientation as the camera and not in the visual space coordinate system, which has its origin in the user's head or in the `[Camera Rig]`-Object respectively. That means not only that the user's and the camera's z-axis are pointed in opposing directions, but as the cameras are tilted upwards and sidewards to face the eyes, x- and y-axes are not matching either (see Fig. 4.1). Instead of working out a transformation matrix to transform the aforementioned normal vectors, it was decided to rather concentrate on a robust Pursuits implementation for the virtual environment and to take calibrated gaze data. Pursuits does not relate on the correctness of gaze data but only that the direction of pupil movement resembles the direction of object movement. This means that only one initial or occasional calibration suffices, which is still an advantage over having to calibrate every time the headset is put on.

The two technologies, Pupil Labs eye tracker and HTC Vive, can now be united in one Unity scene, using the `PupilGazeTracker` and `[CameraRig]` objects. The system described in the following sections manages the tracking of objects that are bound to the user space, as mentioned in section 3. Within Unity this means that all trackable objects have to be child elements of the `[CameraRig]` object and the coordinates that are correlated are linked to the coordinate system of this object and not to the world coordinate system. For the purpose of making objects selectable by gaze, several classes were created of which the most important ones are described in the following paragraphs.

`PupilGazeTracker`: The class was modified to serve as a global timer. As Pearson correlation of movement finds out if two objects are changing their position at the same rate over time, a consistent datum for timing alignment was needed. The Unity Engine itself provides a `Time` class fulfilling all needs of setting timestamps or measuring time spans.

When a script is attached to a Unity object, there are several Event Functions it can implement, which then get called by the Unity Engine at certain events. For example, `Start()` is called once when the game is started, `Update()` is then called every time a frame is calculated by the graphics adapter. These Event functions have certain privileges, like accessing other objects' information within the game or in general make use of functionality provided by the Unity Engine, like the `Time` class. This means the call to access this information or functionality has to come from an Event Function. The receiving of gaze data is performed within a thread to prevent Unity from blocking while waiting for data on the subscriber socket. The aforementioned event (`OnEyeGaze`) is also triggered from within this thread, thus all the subsequent functions and routines that get called when this Event is fired do not have access to the mentioned basic information and functionality of the Unity Engine.

Therefore the `PupilGazeTracker`-class was chosen to be the clock for all entities in the system to base their time related calculations on. It was amended the global variable `_globalTime`, a

[23] PupilHMDCalibration - GitHub - `https://github.com/pupil-labs/hmd-eyes/tree/master/unity_integration_calibration` (visited 2017-09-06)

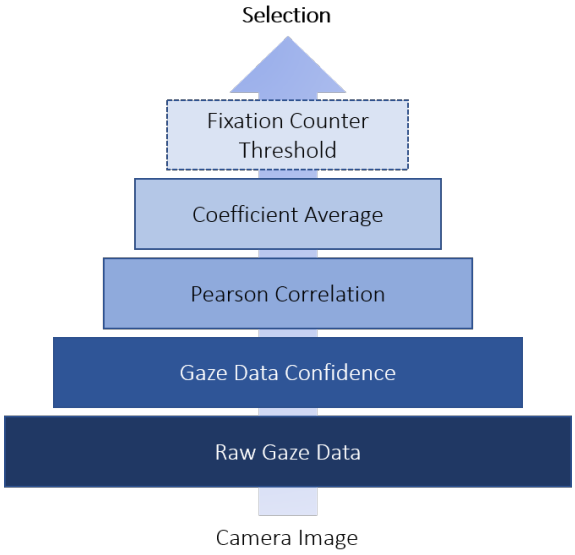


Figure 4.2: Schematic depiction of the filters the gaze data has to pass before it results in a selection. The Fixation Counter is optional.

`System.TimeSpan` that is constantly updated by an extra thread to be the time difference between the current system time `System.DateTime.Now` and the time point when the connection to Pupil Capture was established.

When a gaze data package is received, it represents the position the user looked at at the moment the packet was created by the Capture software. Until the time it is received, some time has passed and the user is already looking at another position. Depending on the setting, this time difference can be negligibly small or negatively effect the Pearson correlation. For this reason, every time a gaze data packet is received, the round trip delay is calculated by sending a request to Pupil Capture and measuring the time until the answer is received. Half of this value is saved together with the x- and y-coordinates of the contained gaze point in a public variable so that other classes can access this information.

`MovingObject`: The `GameObject` class within Unity represents an actual object that exists in the game environment and contains all its information. The `MovingObject` class was created to serve as an envelope class, which besides the actual `GameObject`, contains additional information and functionality (see Listing 4.2). As most of this functionality is also needed to manage the gaze trajectory, the class is designed in a way that can be used for both, objects and gaze. It contains three lists: `movingCorr` is used to store all coefficients calculated for this object within the last `y` milliseconds; `trajectory`, containing the actual coordinates for either gaze or the object within the last `w` milliseconds; and `tpBuffer`, a queue in which new positions get saved while a `MovingObject` object or one of its components gets cloned to prevent changes while being copied. The latter has been introduced because during calculation of the Pearson coefficient by the `Correlator` class, it loops over the contents of the first two lists. If during this loop a new gaze point is received the list gets changed, an exception would be thrown. For this reason and because it significantly reduces lag, calculations are whenever possible performed with copies of MovingObjects. The variables `w` and `y` are also set in the `Correlator` class. A `TimePoint` object pairs a timestamp with a position and a `TimeSample` combines a coefficient with a timestamp.

As mentioned before, additionally to the received gaze point, the `Correlator` class saves the current network delay. These values are used by the `addNewPosition` method that takes the last recorded position of the object from the `trajectory` list and calculates the position of the object

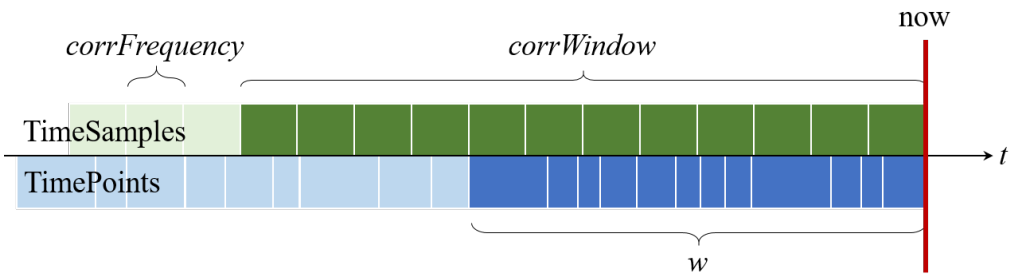


Figure 4.3: **Depiction of relevant data and variables for the calculation of coefficients**: Every white line on the blue bar means that for this point in time the gaze point and the position of each trackable object has been saved. This is initiated by the `OnEyeGaze` event, which gets triggered every time a new gaze point is received. Gaze and position data older than `w` seconds is deleted. Every white line on the blue bar depicts a correlation value that was calculated with this data. Values older than `corrWindow` (or `y` in the `MovingObject` class) seconds are deleted.

at point $t = \_globalTime - timeDelay$ by interpolating between those two positions before saving it as a new `TimePoint`.

The function `addSample` is a vital element of the fixation detection as it calculates the final coefficient that is used to determine whether an object is fixated on or not. It is in the nature of gaze data that it succumbs to jitter, which also affects Pearson coefficients. To even out sudden drops or peaks of the correlation value, a time window `y` can be set within which coefficients get averaged before being processed any further. The mentioned method takes a correlation value, saves it in `movingCorr` and returns the mean of all coefficients within the last `y` seconds (see fig. 4.3).

The class provides some other functions like `cleanUpTraj(int w)` and `cleanUpCorr(int y)`, which remove elements that are older than w or y milliseconds from the respective lists to release memory, logging functionality to log the object's position to a file and a function that colors the object red to signal participants that it is the target object. The aforementioned problem that non-Event Functions cannot access data like the position from a `GameObject`, is solved by the `updatePosition()` method that gets called by the `Update()` method within the `Correlator` class and saves the object's current position in `_current`, a public vector object that is accessible from any method, coroutine, or thread.

In the Use Case scenario of the main study (see 5.2.2), the selection procedure was slightly modified. To improve selection correctness, a `counter` was introduced that counts the number of times the algorithm detected a fixation on a certain object. A `MovingObject`'s counter has to exceed a certain threshold before it gets selected by the `Correlator` class (see Fig. 4.2). First tests of this method showed, that introducing a threshold for the number of fixations reduced the number of false positives that can originate, among other things, in inconsistent tracking data or trajectories with similar properties.

```
public class MovingObject : ICloneable, IEquatable<MovingObject>
{
    GameObject go;

    private List<TimeSample> movingCorr; // list of correlation values within
        the last y seconds
    public List<TimePoint> trajectory { get; set; } // list of positions
        within the last w seconds
```

```
    private Queue<TimePoint> tpBuffer; // buffer for new TimePoints that come
        in while this object is being copied
    public Vector3 _current { get; set; } // current local position of the
        game object
    public int counter = 0; // number of fixations

    public MovingObject(GameObject go, int id, int trial, string path);
    public void startMoving(); // cause object to become visible and start
        moving
    void cleanUpTraj(int w); // if the oldest TimePoint was more than w
        milliseconds ago, remove the oldest
    void cleanUpCorr(int y); // if the oldest TimeSample was more than y
        milliseconds ago, remove the oldest
    public void addNewPosition(int w); // Adds the current position of the
        GameObject to the trajectory
    public void addNewPosition(float timeDelay, int w); // Adds the current
        position of the GameObject to the trajectory and interpolates between
        the last recorded position and the current position to determine the
        position at when the gaze data was sent
    public double addSample(TimeSpan ts, double s, int y); // Adds a new
        Correlation value to the list and returns the average of all
        coefficients within y milliseconds
    public void updatePosition(); // to be called by the Update() method so
        that threads and coroutines can work with GameObject positions
    public void addNewGaze(float timeDelay, Vector3 gazePoint, int w); // if
        this class is used for the gaze trajectory : adds a new gaze point to
        the
    public object Clone(); // return a clone of this object to calculate with
}
```

Listing 4.2: **Excerpt of the** `MovingObject` **class** used for managing the gaze trajectory and the trajectory of each object in the scene. `w` is the time window that is used for the Pearson correlation, `y` is the time window within which correlation values get averaged.

`Correlator`: This class has three purposes: listen for and store new gaze data coming in, keep track of all objects in the scene available for selection and continuously perform Pearson correlations for the gaze data and each object. When being started, the current Unity scene gets searched for objects. If an object should be selectable via gaze, its tag has to be set to "Trackable" in the Inspector view (see Fig. 4.4). It will then be wrapped into a `sceneObject` class and added to the list of SceneObjects, which trajectories will be correlated to the gaze trajectory. There are four values affecting the Pearson correlation that can be set within this class (see Listing 4.3):

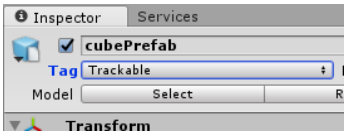


Figure 4.4: All objects with the tag "Trackable" will be taken into account for visual selection

- `w`: Positions and gaze positions that were recorded in the time window $(now - w)$ will be used for correlation.
- `corrWindow`: Coefficients calculated in the time window $(now - corrWindow)$ will be averaged and compared to the threshold.

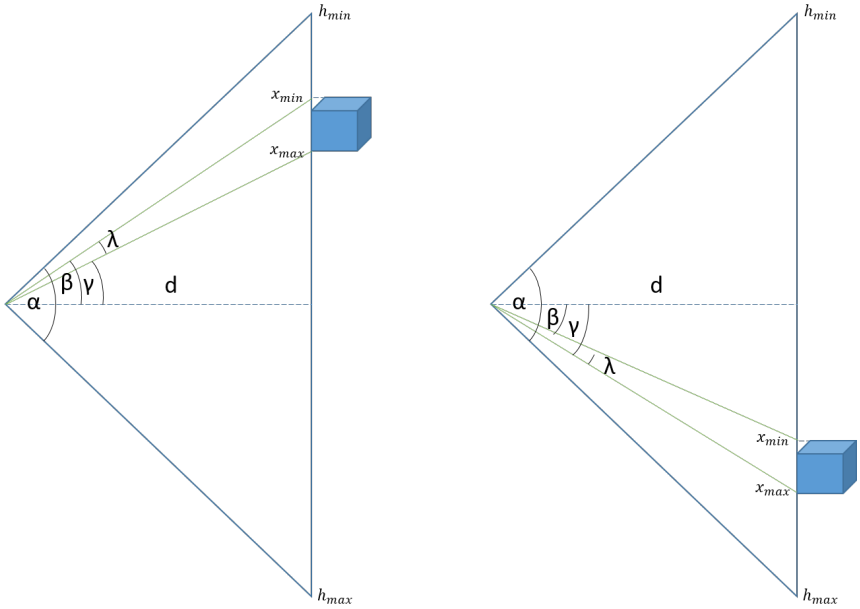


Figure 4.5: Model used for the calculation of the horizontal size of a displayed object in visual degrees. The vertical line constitutes the screen. α: horizontal FOV, β: angle between the object's left edge and the horizontal center, γ: angle between the object's right edge and the horizontal center, λ: visual angle that the object occupies in the user's FOV, x_{min}: minimum x-value of the object in pixels, x_{max}: maximum x-value of the object in pixels, d: distance eye to screen, h_{min}: left edge of the FOV in pixels, h_{max}: right edge of the FOV in pixels.

- `corrFrequency`: The correlation is executed every $(corrFrequency)$ seconds.
- `threshold`: The threshold to which coefficients are compared to. The higher the threshold the more accurate the detection but depending on gaze data quality, a higher threshold may result in longer selection times or even hinder it completely.

These variables can be set prior to starting the game. If there is a script that creates a `Correlator` object, e. g. in the course of a study procedure, the `waitForInit` flag has to be set to true. Thereby, all variables can be set before additional objects and coroutines are created that also use these values. When all values are set, the `Init()` method has to be called, which, among others, handles the registering of objects and subscribing to the `OnEyeGaze` event. Every time this event is fired, every `MovingObject` is induced to record its current position (`addNewPosition`, see Listing 4.2) and the current gaze point is saved to `gazeTrajectory`. This way it is ensured that all data is recorded at the same point in time. At the end of the initialization process, which also includes creating logging files, a coroutine is started that performs the Pearson correlation.

Coroutines are like functions but they have a vital advantage: the `yield` statement. Every time this statement is called, the code from the `Update()` methode can continue hence execution is not blocked. This is convenient if code has to be executed continuously and independently while the game is running, like in this case, to calculate Pearson coefficients. In the coroutine `CalculatePearson` the calculation takes place within a `while` loop as follows: First, copies of all objects are generated to prevent access conflicts during the calculation. Then, pairwise Pearson correlations are performed for x- and y-coordinates separately. In the case where an object is moving solely along the x- or y-axis, the correlation result for the other axis is NaN, as a list only consisting of zero values cannot be correlated. In this case, the value is set to zero. Eventually, the mean of the x- and y-coefficient is calculated and stored in the respective `MovingObject` using the `addSample` function (see Listing 4.2) and its return value is stored in

a list of results. If the highest value from this list is greater than the preset Pearson `threshold`, the corresponding object counts as fixated. At the end of the while loop, the time t_{corr} needed for the whole calculation procedure is measured and by using the command `yield return new WaitForSeconds(corrFrequency-tcorr)` the coroutine halts an amount of time so that the next calculation begins `corrFrequency` seconds after the last one was started.

When a fixation is detected, its position, size and other information about the current state is written to a logfile. The size is logged twice, once in Unity units and once in visual degrees. The calculation of the actual size of an object in the field of view is performed in the `VisualDegrees` class and the procedure is described in section 4.4. In the Use Case Scenario (see section 5.2.2), the `Correlator` class is modified in a way that every time a fixation on an object was detected, its `counter` value is incremented. Subsequently, the object which counter exceeds a predefined threshold is selected (see Fig. 4.2).

`CircularMovement`: A simple class that can be attached to any `GameObject`. It causes this object to move on a circular trajectory with given radius and velocity around a distinct axis with its original location as center. Throughout the studies carried out in section 5, objects always rotated around the z-axis, i. e. on an XY-plane perpendicular to the camera's z-axis.

```
public class Correlator : MonoBehaviour {
    public double threshold; // Pearson threshold

    // w: time window for the correlation algorithm, corrWindow: time window
        in which correlation coefficients are averaged
    public int w, corrWindow;
    public float corrFrequency; // defines how often the correlation takes
        place

    List<MovingObject> sceneObjects; // list, in which all trackable objects
        in the scene are stored
    MovingObject gazeTrajectory; // MovingObject, in which the gaze
        trajectory is stored

    public int lookAt = 0; // which object is the participant told to look at?
    public volatile bool _shouldStop; // if true the Correlator exits all
        procedures cleanly without errors
    public bool waitForInit = true; // set this to true when the Correlator
        object is created by script to allow further amendments before
        routines are started
    VisualDegrees vd; // used to calculate the actual on screen size of an
        object in visual degrees
}
```

Listing 4.3: **Excerpt of the `Correlator` class**

For the further course of the thesis, w is used to described the correlation window length in milliseconds, w_c stands for the coefficient averaging window length in milliseconds, th is the Pearson threshold and f the duration for one correlation cycle in seconds.

4.4 Measurement of the Field of View Wearing a Virtual Reality Headset

In order to report sizes and distances investigated in the main study, a method was needed to transform Unity units into visual degrees. In conventional eye-tracking studies that involve interaction

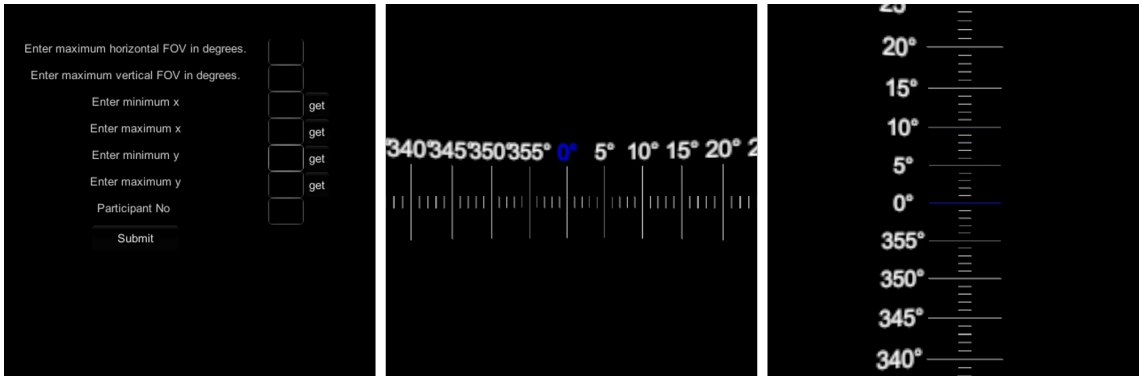


Figure 4.6: **Left**: Input form for the acquired data. The "get" buttons convert the current x- or y-position of the vertical or horizontal red line into pixels and enter the value in the corresponding box. The "submit" button writes the measurement data to a CSV file. **Middle**: Horizontal scale that was presented to the users in order to determine their horizontal FOV. **Right**: Vertical scale that was presented to the users in order to determine their vertical FOV.

with a screen, the person is positioned in a spot at a certain distance from the screen. Using this distance, the size and the resolution of the screen, visual angles can be calculated for the displayed objects. But due to different skull shapes the distance between the eyes and Vive lenses varies from person to person. Therefore, an object with a certain size has varying values in visual degrees. At the beginning of each study session the participant was shown a Unity scene in which the individual values for the total available horizontal and vertical field of view were measured and matched to the corresponding pixel values. This procedure consists of four steps:

First a horizontal scale is displayed that consists of 360 lines evenly distributed on an XZ-plane around the user's head in the virtual environment. Every five lines the corresponding value is displayed so that the participants can tell it to the experimenter. All scale elements are child elements of the virtual Vive camera in Unity so that the scale moves in the same manner as the user's head and its position in relation to the user's eyes remains constant. The scale is then rotated around the y-Axis so that eventually the line denoted with zero degrees meets the left edge of the screen area the user can perceive through the lenses. Now the user is asked to read the value off the scale at the very right of his field of view which constitutes the horizontal field of view the user has wearing the headset (see Fig. 4.6). In the second step, the vertical field of view is determined in the same manner by positioning a 360° scale on a YZ-plane around the user's head.

Subsequently, it is determined to which pixel values the minima and maxima of the field of view correspond to. This is simply done by moving a vertical red line in front of the user until it meets the left and right edges of the available field of view and by recording the corresponding x-values in pixels. This value can be computed by using the `Camera.WorldToScreenPoint` method. It converts the world coordinates (as opposed to local coordinates) of an object into x- and y-values for the corresponding pixels on the screen. Accordingly, the same procedure is performed with a horizontal line for acquiring the individual y-values. The gathered user data is saved to a CSV file and can be accessed by any other scene by searching for the distinct user ID. Given these six values, sizes, positions and distances in degrees can be calculated using the Law of Tangens (see Fig. 4.5).

$$d = \frac{W}{\tan\left(\frac{\alpha}{2}\right)} \tag{2}$$

$$\beta = \arctan\left(\frac{\left|x_{min} - \frac{W}{2}\right|}{d}\right) \tag{3}$$

$$\gamma = \arctan\left(\frac{\left|x_{max} - \frac{W}{2}\right|}{d}\right) \tag{4}$$

$$\lambda = \begin{cases} \beta - \gamma & \text{if } x_{min} < \frac{W}{2} \wedge x_{max} < \frac{W}{2} \\ \beta & \text{if } x_{min} < \frac{W}{2} \wedge x_{max} \equiv \frac{W}{2} \\ \beta + \gamma & \text{if } x_{min} < \frac{W}{2} \wedge x_{max} > \frac{W}{2} \\ 0 & \text{if } x_{min} \equiv \frac{W}{2} \wedge x_{max} \equiv \frac{W}{2} \\ \gamma & \text{if } x_{min} \equiv \frac{W}{2} \wedge x_{max} > \frac{W}{2} \\ -\beta + \gamma & \text{if } x_{min} > \frac{W}{2} \wedge x_{max} > \frac{W}{2} \end{cases} \tag{5}$$

For better readability $W = \frac{h_{max} - h_{min}}{2}$ applies to equations 2 to 5. With equations 2 to 4, the values for β and γ are computed. The actual width in degrees that an objects occupies in the user's field of view can then be calculated by either adding or subtracting these values as described in equation 5. With the same procedure, the vertical visual angle is determined.

5 Evaluation

The studies described in this section were designed to investigate the applicability and performance of the system developed in section 4, which is able to trace fixations on any user-bound object that moves in the field of view. Furthermore it is to be determined how object parameters like size or movement influence the performance and to derive guidelines that help designing a performant, reliable Pursuits interaction in virtual reality.

5.1 Trial Study

The trial study had several purposes: first, to investigate object parameters in an isolated manner and to identify the ones which influence should be further investigated in the main study. Second, to exclude certain settings, which do not work and therefore do not need to be investigated. Third, to find parameters for the correlation procedure that result in reliable selection. The trial study was conducted with 14 participants. Measurements are indicated in Unity units which was changed in the main study using the procedure described in 4.4. One Unity unit corresponds to one meter in "real-world units". Objects moving on trajectories referred to as "horizontal" only alter their x component and objects on "vertical" trajectories only alter their y component of their local position respectively. In both cases, the objects moved along the whole available visual range.

5.1.1 Setting

The virtual objects were decided to be numbered cubes to easily tell the participants what to fixate on. All objects were located on a plane that was orthogonal to the camera's z-Axis and located at a fixed distance of 6.78 units in front of the camera. The available visible space was determined empirically and reached from -3 to 3 units both horizontally and vertically given this distance. When a fixation was detected by the software, the respective object was highlighted with a halo effect. There were several scenes created to investigate four different variables. In the following enumeration the scene is referenced with its number and the independent variable that was being focused on. The scenes are depicted in figure 7.7.

Scene 1a | *Number*: A cube with an edge length of 0.7 units was moving back and forth on a horizontal trajectory that ranged from one edge of the visible space to the other, thus on a trajectory with a length of 6 units. To determine up to which number the system can correctly differentiate which object is selected, $\{3,5,7,9\}$ cubes were shown at the same time. At each increase of the number of displayed objects, two trajectories were added to the top and bottom of the existing ones. Each cube started moving from a random position on its trajectory. The trajectories had a vertical distance of 0.7 units to each other so that they spread out evenly over the visible area. As the Pursuits approach would not work if all objects had the same velocity, they were all moving with unique speeds. The minimum difference between two speed values was 0.2 units per second.

Scene 1b | *Number*: Additionally, circular trajectories were investigated. $\{2,3,5\}$ numbered cubes were moving clockwise along a radial path on the aforementioned orthogonal plane with a radius of 2 units and a velocity of $45°s^{-1}$.

Scene 2a | *Position*: In this scene, three objects were shown at a time at the edges of the visible area to determine if the object position has an impact on tracking performance. The objects moved along horizontal trajectories and were positioned either at the top or at the bottom of the field of view. The vertical difference between them was 1 unit with the outermost trajectory positioned at ± 3 units. The objects started at random positions along the trajectory and had unique velocities with differences of 0.4 units per second.

Scene 2b | *Position*: Three objects on vertical trajectories were positioned at the left or at the right edges of the FOV. The trajectories always had a distance of 1 unit to each other with the outermost trajectory positioned at ± 3 units. The velocity delta was 0.4 units per second to make differentiation possible.

Scene 3 | *Size*: Cubes of three different sizes were compared to each other to find out if object size influences detection performance. Three cubes were displayed at a time on horizontal trajectories with an edge length of $\{0.3,1.5\}$ units to compare them to the standard 0.7 unit cubes. Their trajectories had a distance of one unit in the small and 2.5 units in the large case to prevent overlapping. Their velocities differed by one unit per second.

Scene 4 | *Trajectory Shape* : Three different trajectory shapes {horizontal, vertical, circular} are compared to each other within one scene each being equipped with a 0.7 unit cube. The linear trajectories had a length of 6 units and a speed of 2 units per second, the circular trajectory had a radius of 2 units and a speed of 45 degrees per second.

5.1.2 Procedure

At the beginning of the experiment, each participant was shown how to put on the headset. Then, it was adjusted in a way that it was attached safely but not too tight to their head, using the three Velcro fasteners of the Vive's harness. In every scene, participants were asked by the experimenter to fixate on a certain object by naming its digit out loud. The experimenter then controlled visually if the mentioned cube was being highlighted. If highlighting of an object was not constant, wrong or not given at all, detection quality was tried to improve by manipulating the values for the correlation process: correlation window w, averaging window w_c, correlation frequency f and pearson threshold th. If many false detections were registered, the th was increased. However if no detections occurred it was decreased. If detection was correct but not constant, both w and averaging window w_c were increased. Equally, a balanced setting for f was searched by increasing frequency if detection lag occurred and decreasing it in case of rendering lag. After each setting the users were asked which characteristics they found most convenient and represented best what they were looking at. The answers and other remarks were documented in a Google Spreadsheet.

5.1.3 Results and Discussion

During the course of the trial study, qualitative data has been gathered which are discussed in this section. Most reliable results were achieved with Pearson threshold $th = 0.4$, a correlation frequency $f = 3.33$, a correlation time window $w = 300ms$ and a correlation averaging window $w_c = 900ms$, which means that the average of the three last coefficients for an object was compared to the threshold (also see section 4.3).

All participants but one preferred circular over linear trajectories, because they were more convenient to follow and showed the best detection performance with is in line with the findings from [18] and [50]. One participant mentioned that this was due to the larger distance between the objects. Furthermore, all but one person preferred smaller cubes over large cubes. Most of them agreed on them being easier to follow and to focus on. When it came to the objects' positions many agreed that differences between top and bottom (respectively left and right) were hard to tell. When they had to decide which position was more convenient, there was no discernible tendency except least users voted for the upper position. Regarding the number of objects, most experimentees preferred no more than five objects. As for the linear trajectories, this was to the perceived drop in detection performance. Because there were hardly any detection errors with five objects on a circular trajectory three more cubes were shown additionally moving counter-clockwise to sound out the limits of this setting. Although detection was still reliable, most agreed on a limit of five objects due to visual clutter.

As there was visual feedback on the selection provided the participants knew if a selection was right or wrong. If the software detected fixations in one setting better than in another this might have had an influence on their opinion.

5.2 Main Study

The main study had three primary goals: first, to investigate how object and trajectory parameters influence performance of the developed algorithm in order to derive design guidelines for traceable objects in a virtual reality setting employing Pursuits. Second, to determine to which amount pursuing objects is possible while performing another task that requires visual attention. Third, to demonstrate use cases in which Pursuits are employed and gather qualitative feedback on this form of interaction. It consisted of two main scenarios, each subdivided in one "stationary" and one "moving" setting. The aim of the latter is to ascertain to which amount performance changes compared to the static setting and therefore, if Pursuits is a suitable input mechanism for settings in which the user is navigating. During the study procedure every participant passes through every setting in the following order:

I **Abstract Scenario:** For evaluating the performance of the employed detection algorithm participants had to fixate certain objects, once while being seated (setting no. 1), once while moving (setting no. 2) (see section 5.2.1).

II **Use Case Scenario:** Two scenes were created to demonstrate and investigate the developed interaction technique in different use cases (see 5.2.2). In one scene, participants had to interact with an ATM machine (setting no. 3) while standing or being seated. In another scene, they had to shoot asteroids in an outer space scenery while simultaneously moving around the room (setting no. 4).

The following three assumptions about detection performance were to be confirmed or disproved:

1. There is a drop in detection performance while moving compared to a static setting, due to increased mental load.

2. Smaller objects offer a smaller area in which fixations can occur, which results in an increased detection performance.

3. Pursuing trajectories with larger radii while walking is hindering the main navigation task and thus a decreasing detection performance, as they are closer to the outer limits of the field of view.

The `Correlator` settings found credible in section 5.1.3 were used throughout the main study. Before each session, the field of view of each participant was measured using the method described in section 4.4. Values for object and radius sized can be found in Table 5.1 where results of these measurements are summarized. Figure 7.8 shows screenshots of all four scenes described in this section.

5.2.1 Abstract Scenario

The abstract scenario aimed for investigating the influence of certain object and trajectory parameters on detection performance to acquire quantitative data for further analysis. There should be no biasing clues or embellishments distracting participants from their task and to make sure the observed effects are solely due to manipulations of the independent variables. For this reason, the

Clear Flags[24] setting was set in a way that all screen parts but the objects to be coloured black. As the relative size of an object on the screen is a monocular depth clue, a script was created to ensure objects always have the same absolute size to the viewer, regardless of their distance [60]. An object will be down-scaled when gets closer the viewer (decreasing z-Value) and up-scaled when it moves away (increasing z-Value). Another key difference to the Use Case Scenario is that no feedback is provided if or which object is detected, which might have an impact on subjective ratings.

As the *position* did not have a mentionable effect and circular *trajectories* were rated both most reliable and comfortable to use in the trial study, these variables were decided to be fixed throughout the main study. Furthermore, as detection was rated most agreeable with up to five objects, this was the final choice for the number of objects per trajectory.

- **Independent variables**: Trajectory radius $\{0.5, 1.5, 2.5\}$, Object size $\{0.3, 0.65, 0.9\}$, Object Depth $\{2, 6, 16\}$, User is moving $\{yes, no\}$
- **Dependent variables**: Selection time in seconds, selection correctness $\{correct, notcorrect\}$
- **Fixed variables**: Object velocity $\{45°s^{-1}\}$, trajectory position $\{center\}$, trajectory shape $\{circular\}$, No. of objects per trajectory $\{5\}$

There are three independent variables with three characteristics respectively, and one independent variable with two characteristics; therefore, 54 different conditions are shown to each participant (within-subject-design). These have been subdivided into two scenes: participants undergo the 27 different conditions resulting from all combinations of *radius*, *size* and *depth* once being seated and once while walking ("Static User" and "Moving User").

In each condition, there are five colored cubes of the same *size* moving along one circular trajectory with a given *radius* at a given *depth*. One cube is randomly chosen to be coloured red, the other four ones are coloured in light blue. The random selection is necessary to achieve a constant condition as prior knowledge of where the target is going to appear would decrease fixation time (i.e. the time until the first saccade is completed) [49]. The trajectory centre is in the centre of the participant's field of view. The objects appear and start moving when the experimenter presses the space bar. The objects disappear as soon as a cube is selected whereupon the next setting is prepared with the cubes not yet visible.

The selection time is measured by the system and describes the time-span from the objects appearing until one of them is selected. A cube is considered selected by the software as soon as its coefficient is greater than the others' and exceeds the predefined threshold (the procedure for calculating the threshold is described in section 4). For identification purposes the cubes are numbered and have their respective number as a black digit texture appearing on each of their faces. A selection is considered false if a blue cube is selected. Both variables, selection time and selection correctness are logged and saved into a file by the software.

After each of the two scenes, participants had to answer a questionnaire with general questions on the absolved tests and a NASA-TLX questionnaire (without pair-wise comparisons, available either in English or German) to measure the perceived workload while performing the respective tasks [61] [62] [63]. This was done once after each of the two scenes.

[24] A Camera Object in *Unity 3D* has a field called 'Clear Flags' which determines what is happening in screen parts in which no objects are drawn. E. g. setting it to 'Skybox' results in drawing a predefined spherical surrounding, such as a cloudy sky with a skyline. The setting 'Solid Color' allows for picking a color that is drawn over those unused parts.

Static User

The seated user is shown every combination of size, radius and depth in a black surrounding in a randomized order and is told to fixate on the red cube until all cubes disappear. To further reduce stimuli, the Vive's play area visualization is disabled in this scenario. After the scenario participants filled in form 7.1.

Moving User

To evaluate selection performance while the user is moving, the participant is told to perform a simple walking task. A white square on the floor indicates where the user is supposed to go. The square alternately appears in the middle of two opposing borders of the virtual play area and changes its position each time the user comes within a certain range of its location. For safety purposes bounds were visible in this scenario. Additionally, before this trial the participants were told to walk up to the bounds and to familiarize with them by feeling for the walls in the real environment. Time and error rate measurements are performed in the same way as in the static task. Thereafter, participants filled in form 7.2.

In the original study design, the white square appeared in random locations within the available area. After some test trials it appeared that the process of searching end re-orientating was much too demanding and resulted in the users feeling overwhelmed and being tangled up in the cord of the headset.

5.2.2 Use Case Scenario

This scenario is aimed at examining the applicability of Pursuits in concrete use cases. Two scenes were created: one in which participants are required to be moving and one in which they are not. In both scenarios users had to fixate on user-bound objects moving along a circular trajectory to trigger certain actions in order to achieve a goal. The participant is provided input feedback on the correctness of the selected objects. At the end of each setting, qualitative feedback was gathered with both a NASA-TLX questionnaire and a User Experience Questionnaire (UEQ, see 7.5). The latter was employed to ascertain the overall impression the users received of the presented interaction technique. The UEQ is broadly used e.g. in evaluations on user satisfaction or quality assurance during software development. It consists of 26 scales in form of a seven grade semantic differential[25] [64] [65]. The questionnaire provides differentials for six different categories: attractiveness, perspicuity, efficiency, dependability, stimulation and novelty. Each category is represented by four different scales but attractiveness, which has six scales.

Static User

Wearing the headset, the participants were asked to stand in front of a virtual ATM machine. Ten cubes moved in front of them, each of which was tagged with a number between 0 and 9. The task was to enter a random four digit PIN given by the experimenter by fixating on the respective cube. Numbers one to five were moving in a clockwise manner along one circular trajectory located in the left half of the visual space, numbers five to nine and zero were moving counter-clockwise on a second trajectory located in the right half. The cubes had a width of 0.5 units and moved on a circular trajectory with radius 1.5 at an angular speed of $45^\circ/s^{-1}$. Every time the correlation between a cube's trajectory and the gaze trajectory results in a coefficient that is above the Pearson threshold, the software increases an internal counter. If the counter exceeds a

[25] A *semantic differential* gives experimentees the possibility to express the degree they associate something (in this case: interacting by pursuing) with certain characteristics. This is done by presenting them couples of opposing adjectives. In this case they cannot just choose the one that they think applies best but show their degree of accordance on a seven grade scale.

predefined threshold, the number on the cube is considered as the intended input. For the static setting, the counter threshold was set to 20. Besides that, the same algorithm as in the abstract scenario was used with the same parameters (see figure 4.2). When an input was performed, the user gets feedback by a light flash, the signal to proceed with the next digit. If after ten seconds none of the counters exceeds this threshold, the software records an error for the current digit and proceeds. If needed, some trial entries were performed, to get the participant accustomed with the procedure. During the PIN entry, the screen of the virtual ATM machine gave feedback on the progress, by showing a number of dots that corresponded to the number of digits that were already entered. After the fourth PIN was entered, the screen showed whether the PIN code was correct or false.

Moving User

For the moving user task, a game was created, in which the user is asked to "blast" asteroids[26] that are user-bound and move within the field of view in an outer space setting. The target asteroid is colored red while the other objects are colored blue. There is only one target object visible at a time to clearly detect wrong selections. The centers and radii of the object trajectories as well as the meteor sizes, rotations and angular velocities are randomly chosen by the software. The radius was selected from within a range of $[0.5, 2.5]$, angular velocity from a range of $[25°s^{-1}, 90°s^{-1}]$, trajectory center from a range of $(xyz) : ([-3,3][-3,3][5,11])$ and object scale from a range of $[0.5, 1.5]$. These limits were set so that objects do not always have the same appearance and that they mostly move within the visible area in front of the user and only sometimes leave that area. The latter has been decided to investigate if users are able to imagine where an object should be after it left the field of view until it re-enters. To avoid overlapping, if a collision is detected when an object is created, it gets deleted and re-initiated at another position. The same detection technique as in the stationary scenario comes into effect, i. e. an asteroid gets selected, if its counter exceeds a certain threshold. Feedback on the selection is given by an exploding effect of the respective object combined with the playback of an explosion sound effect[27]. Every second, the software ensures that there are ten asteroids in the scene and that one of them is set as the target. During the trial, the experimenter had the possibility to adjust the counter threshold: if a participant performed many wrong entries, the threshold was increased to increase detection correctness, if many correct entries were achieved, the threshold was lowered to reduce input time. The resulting different input times can be used to determine, if duration had a negative effect on subjective ratings. The initial counter value was 15.

While the participants are "shooting" asteroids, they have to walk within the available space. Like in the abstract scenario, a white square on the floor appears alternately at two opposing edges to indicate where to go. After the game is explained to the participant, it is started by the experimenter and runs for 180 seconds.

5.2.3 Results

During the study it became apparent that the eye tracking software could not track all eyes equally well. In case of poor tracking, adjustments were made, like justifying the camera focus, defining an area of interest in the camera image to point the software to where the pupils are or recalibrating the trackers. In one case, the participant's make-up seemed to mislead the algorithm into detecting a pupil in the eyelashes (see fig. 5.1). Participants were asked to take off their glasses if possible as this has shown to improve tracking performance. In two cases this was not an option due to strong

[26] Objects are generated using the 'Asteroid Pack - by Pixel Make' package by Pixel Make available at the Unity Asset Store (https://www.assetstore.unity3d.com/en/#!/content/83951)

[27] Explosion effects are generated using the 'Explosion System' package by Vadim Gerc, available at the Unity Asset Store (https://www.assetstore.unity3d.com/en/#!/content/76511)

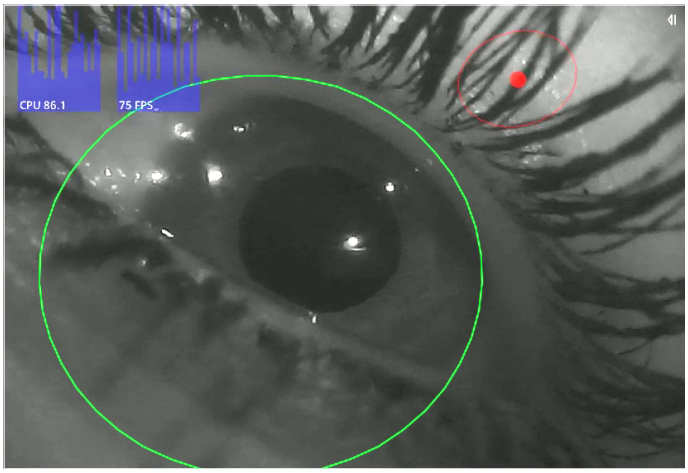


Figure 5.1: Camera image from Pupil Capture recorded during the study. The software seems to detect the pupil (red dot) within the eyelashes of the participant. The green circle indicates where the eyeball is assumed.

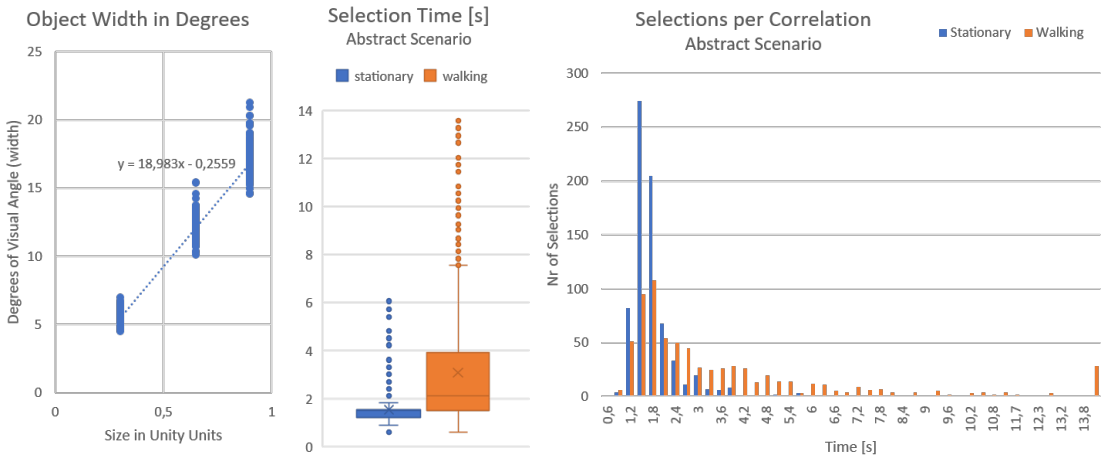


Figure 5.2: **(left)** Graph depicts the dispersion of actual size in visual degrees for Unity sizes {0.3,0.65,0.9}. See table 5.1. Data of one participant was removed, as the procedure for detecting the field of view was not performed correctly.

Figure 5.3: **(middle)** Graph depicts the dispersion of selection durations in the abstract scenario. 25 data points with durations > 15s were removed.

Figure 5.4: **(right)** Data distribution for measured selection times for stationary and walking scenarios. Stationary data has its a peak at 1.2s, walking at 1.8s but is more evenly distributed over time.The outliers $>$ 15s are visible on the right. The data is strictly positive and positively skewed. Therefore, a gamma distribution with log-link was selected for building the generalized linear model.

Variable	Unity Size	Range $[min, max]$	Mean $[min, max]$	SD $[min, max]$	Mean total (SD)
Sizes	0.3	[4.51°,7.00°]	[5.11°,5.98°]	[0.33,0.41]	5.55°
	0.65	[10.13°,15.43°]	[11.49°,12.95°]	[0.72,0.80]	12.22°
	0.9	[14.58°,21.31°]	[16.44°,17.84°]	[1.05,1.15]	17.14°
Radius	0.5	[4.30°,5.66°]	–	–	4.84° (0.32)
	1.5	[12.71°,16.57°]			14.25° (0.91)
	2.5	[21.47°,26.40°]			22.95° (1.36)

Table 5.1: **Size Table** correlating Unity units to degrees of visual angle. The range column (3) contains the absolute minimal and the maximal value for the respective size and radius of all participants. Column 4 and 5 show means and standard deviations for all minimum and maximum sizes that stem from perspective distortion. There are no values for diameter as they are not subject to distortion. Column (6) "Mean total" contains the mean of minimum and maximum values of visual angle for the respective variable. Values in degrees describe the object width.

shortsightedness. Data of two sessions hat to be removed from the dataset as the participant's eyes were tracked too poorly and the session was aborted, one of which wore glasses. In both cases, Capture could not build a stable eye model, although the aforementioned measures were implemented, and thus did not provide enough gaze data for Pursuits to function.

Another problem was the harness that attaches the headset attached to the user's head, as it could not be tightened enough for two participants. These participants had to wear a beanie throughout the session. Nonetheless, even when the headset fit tightly, it was shaking while the experimentee was walking due to its weight, leading to poor tracking results.

As stated in section 4.3, calibrated gaze data was used as input for the algorithm, which is adjusted to one specific user. Even if the trackers are not calibrated correctly the Pursuits approach works for other users as well, as long as the gaze data at least resembles the object movements. If one participant's physique was completely different from the last one, gaze data did not sufficiently match the eye movements any more and the eye trackers had to be recalibrated. The necessity for this step could be determined during the first scene: if the experimenter realized that no or only few selections were correct, a new calibration was performed and the scene was restarted. This also had to be done if calibration data was lost, which occurred occasionally.

In the abstract scenario, the focus was on "stress-testing", the implemented Pursuits algorithm and on acquiring quantifiable measurement data. In contrast, the use case scenarios aimed at presenting possible applications and gathering qualitative feedback on how this form of interaction is perceived. Therefore, the results section is divided into the presentation of *A. Quantitative Results* and *B. Qualitative Results*, each of which is then subdivided into results from the abstract and the use case scenarios.

A. Quantitative Results

In section 4.4, a procedure was described to calculate the actual visual angle an object occupies within the user's field of view. This procedure was performed with every participant at the beginning of the study. Object and radius sizes investigated in the Abstract Scenario resulted in different visual angles, depicted in Fig. 5.2. The mean values for object sizes $\{0.3, 0.65, 0.9\}$ are $\{5.55°, 12.22°, 17.14°\}$ and Unity units for radius sizes $\{0.5, 1.5, 2.5\}$ resulted in mean values $\{4.84°, 14.25°, 22.95°\}$ (see table 5.1). The mean horizontal field of view was 69.56° ($SD = 6.16$).

A.I Abstract Scenario

First, a general data exploration was performed. 26 Participants performed a selection for each combination of three distinct values for radius, size and depth respectively, while sitting and while walking. This results in a total of $3 \cdot 3 \cdot 3 \cdot 2 \cdot 26 = 1404$ measurements. When participants walked while fixating on the objects, some extreme values were recorded for the **selection time**; therefore some data points had to be excluded. It was decided that every selection that took longer than 15 seconds is to be excluded for the analysis of the selection time. In a real-world setting, any selection time longer than this would be impracticable, which is why these cases were marked as errors, regardless of correct or incorrect. With this procedure, 29 data points were identified of which all were recorded during the walking scenario. 27 of these cases were recorded when a constellation with $radius = 0.5$ was shown. See figure 5.3 for distribution of the remaining data points. The overall mean value for selection time was $3.07s$ $(SD = 1.91)$. Mean values for different depths and sizes do not differ much from each other, but there is a decline in mean times recognizable for an increasing radius: the radii $\{0.5, 1.5, 2.5\}$ resulted in mean selection times of $\{2.8s, 2.26s, 1.87s\}$. When differentiating between walking and sitting, it can be observed that this declining effect is relatively small in the first, but considerably big in the second case. The mean selection time while moving was 3.85 seconds long, much longer compared to the stationary setting in which a selection was performed at an average of $1.53s$, which is a ratio of 2.52 (see Fig. 7.9).

The recorded data for selection time is strictly positive and also positively skewed instead of normally distributed, so a Generalized Linear Model was built using SPSS Statistics to investigate the main effects of the independent variables and the pairwise interaction effects between movement and object properties. A gamma distribution was assumed to match the data and selected for the model and a log-link function was applied (see Fig. 5.4). The results confirm a significant effect of the radius in the walking scenario: while in the stationary setting, neither a radius change from 0.5 to 1.5 nor to 2.5 resulted in a significant change in selection time ($p_{\text{radius1.5}} = .735$, $p_{\text{radius2.5}} = .229$), an increase of the radius while walking significantly reduced expected selection time by factors $exp(B)_{walking*radius1.5} = .716$ and $exp(B)_{walking*radius2.5} = .546$ ($p < .001$). In general, the change from stationary to walking setting resulted in an increase of the estimated selection time by a multiplicative factor of $exp(B)_{\text{walking}} = 2.40$ $(p < .001)$. Results show no significant effects for size or depth. For all results see table 7.1.

The mean values for **detection correctness** do not obviously differ within the respective categories (see Fig. 7.10) except movement: while sitting, 75% of all entries were interpreted correctly and while walking only 57% were correct. To find any significant main and interaction effects of the independent variables on correctness, a binomial logistic regression was performed. To take potential differences in the effects of the first three variables under the conditions "walking" and "not walking" into account, pairwise interactions between them and type of movement were integrated in the model. Of the four variables, "movement" had the most impact on correctness and the largest effect was detected for a change from "walking" to "not walking": if the experimentee was moving, an expected odds ratio of 25.4% was estimated compared to the stationary setting. For example, the expected probability for a correct detection under the conditions $radius = 0.5, size = 0.3, depth = 2$ changed from 78.1% while walking by 30,6% to 47.5% if the participant was moving. While the user was stationary, odds radios of near 1 were observed for radius and object size. A mentionable but no significant tendency can be observed for object depth: an increasing depth led to a decrease in detection correctness. The radius had different effects depending on the movement: when participants were walking, a change from $radius = 0.5$ to $radius = 1.5$ results in an estimated odds ratio of $e^{.524+.048} = 1.772$ ($p = .069$), a change to $radius = 2.5$ in $e^{.678} = 1.970$ $(p < .05)$. The deteriorating effect of increasing depth was also observable in the walking scenario, but was not significant either. Object size had no mentionable effect on correctness. For detailed results see table 7.2.

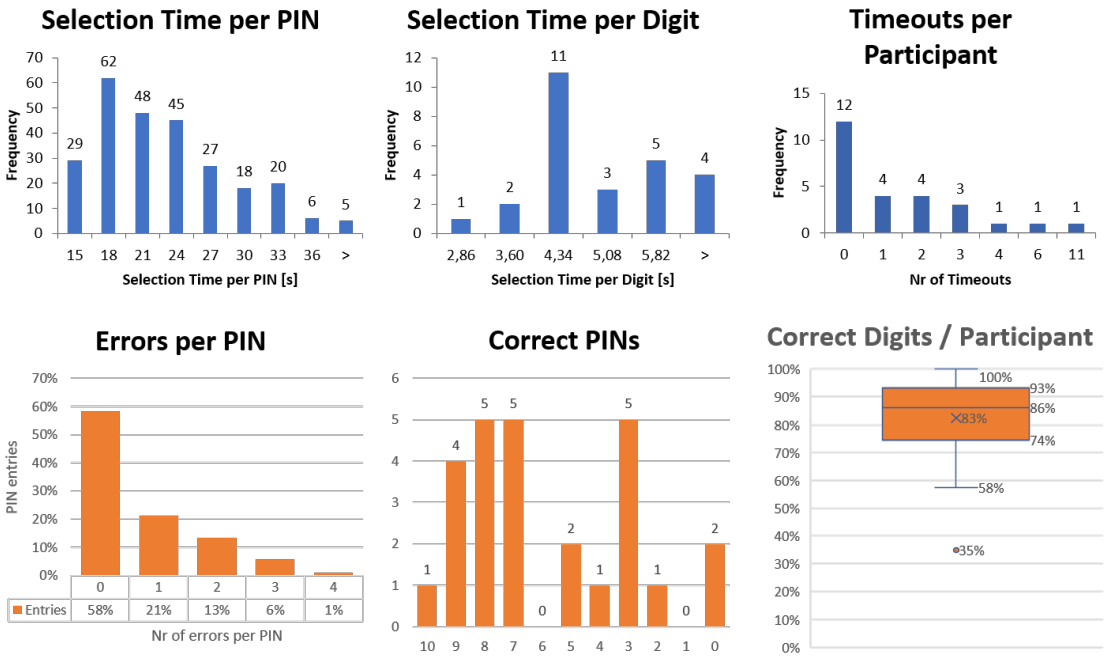


Figure 5.5: **Statistics for Use Case Scenario, Stationary Scene:** the top row covers information concerning selection time; the bottom row covers selection correctness. Histograms show counts per interval, e.g. in the top left histogram, the first column counts selection times [0,15[seconds, the second column of [15,18[seconds and so forth.

As the `corrFrequency` parameter of the correlation algorithm was set to a constant of 0.3 seconds, both correlations and selections were performed with this frequency. Figure 7.11 shows the ratio of correct and false detections for every cycle for stationary and walking setting. Most detections occurred after five cycles, i.e. 1.5 seconds, when sitting and after six cycles (1.8 seconds) while moving. It is obvious that all detections that occurred before cycle three (i.e. below 0.9 seconds) were wrong. Over the next cycles the percentage of correct detections is increasing until it varies around its mean value of the respective scenario.

A.II Use Case Scenario

PIN entry scene Every participant entered 10 PIN codes consisting of four digits, so a total of 260 PIN codes and 1,040 digits were entered. Mean selection time for one PIN was 21.40s ($SD = 6.03$) and for one single digit 4.86s ($SD = 2.11$). If the counter threshold for a successful selection could not be reached within ten seconds, the software recorded an error, which happened 42 times. Out of 26 participants, 12 did not experience a timeout. If only selections without a timeout are taken into account mean values differ slightly: $\bar{t}_{\text{PIN}} = 20.23s$ and $\bar{t}_{\text{digit}} = 4.63s$ ($SD = 5.79$). Mean selection times for a PIN per participant ranged from 13.4 to 32.13 seconds.

During the session, some participants mentioned that they mixed up the numbers 6 and 9. This could later be confirmed in the recorded data. As a consequence, every error that occurred due to this mix-up was deleted. Per participant, on average 83% of digits and 5.84 PIN codes were recognized correctly. See Fig. 5.5 for more details.

Asteroid shooting scene As game time was limited to 180 seconds, the number of selected objects varied between participants. In total 349 asteroids were destroyed, of which 261 were intended. On average, every experimentee shot 13.42 asteroids, of which 73.6% ($SD = 0.16$)

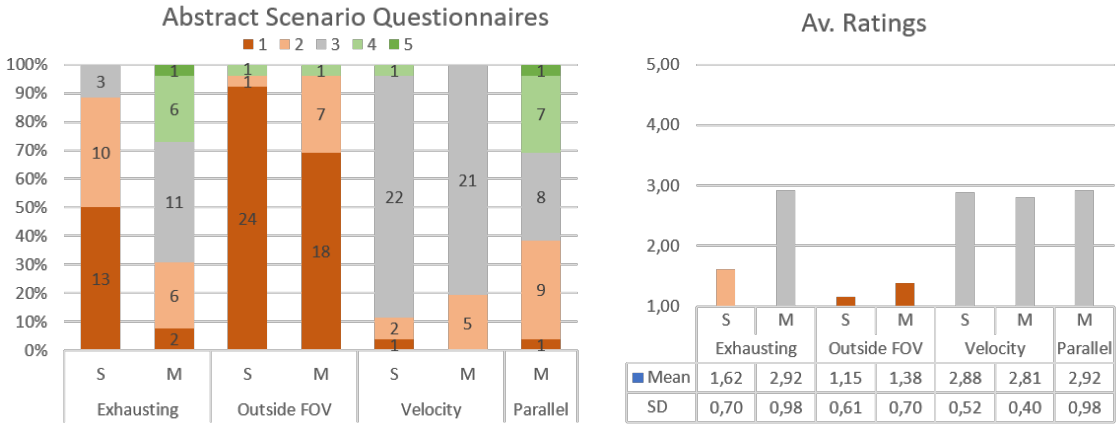


Figure 5.6: **Questionnaire results of the Abstract Scenario** (for the questionnaires see figures 7.1 and 7.2). The graph show how many of the 26 users selected the respective option. Scales for questions 1, 2 and 4 ranged from "fully disagree" (1) to "fully agree" (5). The five step scale for question 3 ranged from "too slow" (1) to "too fast" (5). "S" stands for stationary, "M" stands for moving. The right graph shows means for all questions.

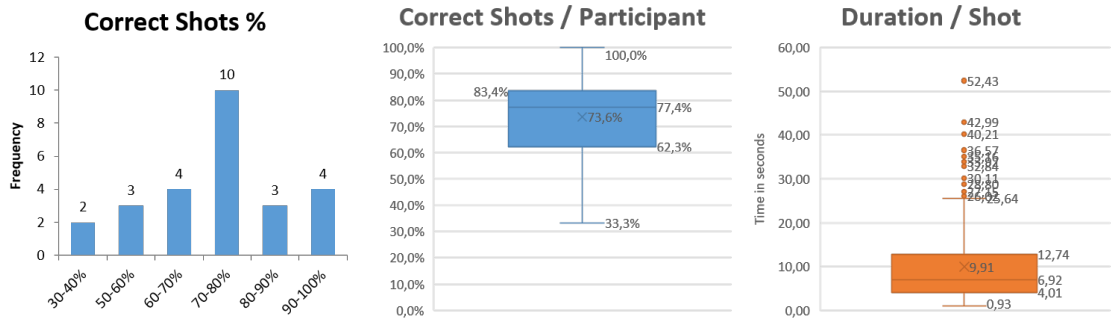


Figure 5.7: **Statistics for Use Case Scenario, Moving Scene**: the first graph shows the number of participants whose selection correctness is within the respective range. The middle graph plots correct selection percentages for all participants and the right graph plots all selection times.

were correct. A selection took on average 9.91 seconds, whereby mean selection times per participant ranged from ranged from $\bar{t}_{min} = 5.67s$ to $\bar{t}_{max} = 19.13s$. Of 26 participants, 2 achieved a 100% correctness. Counter thresholds had been customized for the users during the trial, so its value at the end of the trial represents the detection quality for this individual person. Threshold values reached from 12 to 23. See figure 5.7 for detailed results.

B. Qualitative Results

The 26 participants whose results were analyzed were aged between 21 and 37 ($M = 25.88$, 10 female, 16 male), of which 4 normally wear contact lenses and 13 use no vision aid and 7 normally wear glasses. All of the latter were asked to take off their glasses during the study, as this ensured more stable and accurate tracking. 15 stated that they had used a virtual reality headset 1-2 times before, 4 have already used it more often. 7 participants regularly inform themselves on this topic and 3 were not familiar with the term "VR" at all.

In this section, results from the three different questionnaires are presented: general subjective questions, NASA TLX average scores and User Experience Questionnaire. After each individual scene, participants had to answer a questionnaire with general subjective questions and a NASA Task Load Index. Each scene in the Use Case Scenario was followed by a User Experience Questionnaire and additionally, experimentees were encouraged to express their own thoughts either verbally or through the questionnaire. These results will be presented in the following section.

B.I Abstract Scenario

When analyzing data from the NASA TLX questionnaire two outliers were detected with a total average score delta of $\Delta tlx > 40$ between "walking" and "not walking". It was noticeable that all participants except one rated the walking scenario as more demanding than the stationary one. Due to the small sample size ($N < 50$), a Shapiro-Wilk test was performed to test for normal distribution. The test assessed that the data was not normally distributed ($p < .01$). Taking this into account a paired-samples T-Test was run once for the original data and once with the data transformed by $f(x) = \sqrt{x}$ and the largest outlier ($\Delta tlx = 55$) removed, which led to normal distribution as indicated by a Shapiro-Wilk test ($p > .5$). For both the transformed data ($t(25) = 7.96$, $p < .0005$, $d = 1.56$) and the original data ($t(26) = 5.98$, $p < .0005$, $d = 1.15$, which will be referred to in the further course of this paragraph) a Paired Samples t-Test showed a significant increase in the perceived workload for the walking task. Participants rated the walking scenario more physically demanding (36.38 ± 16.99) compared to the walking scenario (21.07 ± 9.83 [mean $\pm$ standard deviation]). The additional walking task elicited in an increase of 15.3 (.95% CI, 10.04 to 20.56) in the average workload value (See Fig. 5.11).

When comparing results for the question "Focusing on the correct cube was exhausting", which could be answered on a scale between 1 ("fully disagree") and 5 ("fully agree"), the mean of answers for the walking scenario was 2.85 ($SD = .989$), which was significantly higher compared to 1.63 ($SD = .688$), as confirmed by a related samples Wilcoxon signed-rank test ($z = 3.868, p < .0005$). While 25 out of 27 participants disagreed or fully disagreed with this statement, this number decreased to 10 when they had to perform the additional walking task and 7 (fully) agreed that focusing was exhausting. All but one participant gave the second scene a higher rating. Furthermore, opinions on "Focusing on both tasks was easy" were symmetrically distributed. Both 10 people (fully) agreed and (fully) disagreed. For further results and histograms on these questionnaires see Fig. 5.6.

Conclusively to the Abstract Scenario, participants could give additional feedback using the free text section of the questionnaire and verbally. Five users mentioned that, having to orientate, searching for play area boundaries and fixating on the target at once was overwhelming at first but became easier the longer they had to do it and the more they gained trust in the visual representation of the "real-world" boundaries. Three participants found it difficult to orientate and wished that Chaperone bounds would have been visible the whole time and not just when they reached an edge. One participant reported a slight amount of dizziness due to this fact. Two participants noticed the initial moment of only a few frames in which objects were instantiated in the middle of the field of view but not yet at their assigned positions or moving. One of them was irritated due to this phenomenon.

Another effect was observable in the walking task: while at first participants seemed overwhelmed watching the target and searching for the square that indicated the walking direction at the same time, they got more confident over time and could better focus on fixating the target cube as they already know where to head. Some of the participants even mentioned that it got easier over time.

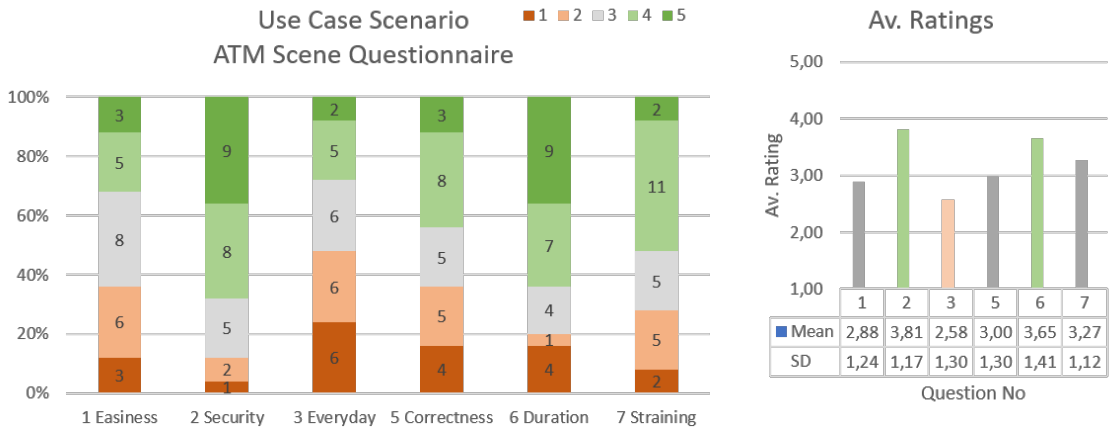


Figure 5.8: **Questionnaire results of the Use Case Scenario, stationary setting**. For the questionnaire see Fig. 7.3. Scales for questions 1-3 and 5-7 ranged from "fully disagree" (1) to "fully agree" (5). The graph on the right shows mean ratings for questions 1-3 and 5-7. For question 4 see Fig. 5.9.

B.II Use Case Scenario

The TLX score differences $\Delta tlx_{\text{(walking-stationary}}$ showed no outliers and a Shapiro-Wilk test confirmed normal distribution ($p = .069$). No significant correlation was found using a Paired Samples t-Test ($p = .962, d = .009$). Participants rated the workload for the stationary scene (47.13 ± 18.45) approximately as high as for the scene, in which they had to walk (47.27 ± 17.04 [mean $\pm$ standard deviation]) (see Fig. 5.11).

To find out if there is a coherence between performance values and subjective rating for both scenes, possible correlations of both selection time and error rate on subjective ratings have been investigated. It was assumed that longer selection times and a higher number of errors result in higher TLX ratings and lower UEQ ratings. Therefore, pairwise Spearman rank-order correlations have been performed on each of these scales.

Stationary Scene: The questionnaire for scene three had seven questions (see Fig. 7.3). 17 out of 26 users thought that entering a PIN by gaze is secure or very secure. However, only 7 participants are in favour of using this procedure in their everyday life, six are indecisive and 13 stated that they would rather not, or not at all, use it. When asked for how often they would be willing to authenticate with the presented procedure, a total of 21 experimentees would consider it, out of which 13 would prefer to use it once a month and eight would use it on a daily basis. 17 users (fully) agreed that the authentication procedure takes too long against, while five thought that is was (rather) not. As during a first trial session it was noticed that the participant tends to face the black void next to the ATM machine to perform the input, participants were asked if they did it the same way. Eleven of them stated that they (mostly) turned away, whereas 14 (mostly) did not.

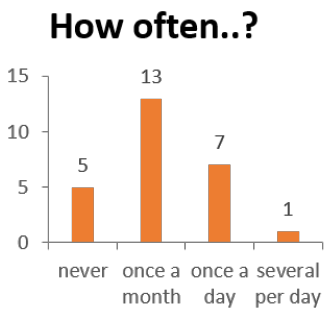


Figure 5.9: Answers for question 4 of the Use Case Scenario, stationary setting

For analysis of the User Experience Questionnaire, form data was transformed to a seven point scale ranging from -3 to 3, whereas adjectives that are generally positively connoted are projected on the highest (3) and negative adjectives on the lowest (-3) score. Values were then averaged by each of the six categories. The PIN input scenario received high mean ratings for the categories Perspicuity (1.31) and Stimulation (1.03) and the highest rating in terms of Novelty

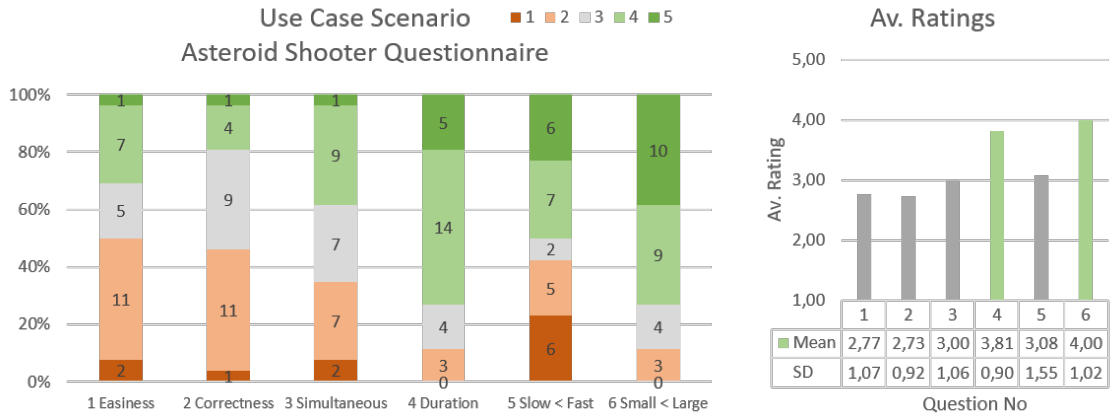


Figure 5.10: **Questionnaire results of the Use Case Scenario, moving setting**. For the questionnaire see Fig. 7.4. Scales for questions 1-6 ranged from "fully disagree" (1) to "fully agree" (5). The graph on the right shows mean ratings for questions 1-6.

(2.06). However, categories concerning Attractiveness (0.51), Efficiency (0.04) and Dependability (0.46) got rated neutrally (see Fig. 5.11).

When looking at the statistics for perceived easiness ($M = 2.88, SD = 1.24$) and correctness ($M = 3, SD = 1.3$) of the input procedure, there is no recognizable tendency. Calculating Spearman's rank-order coefficient reveals that performance had an influence on these ratings in the questionnaire. There are negative correlations for both the number of wrong PIN codes and the average selection time with the perceived easiness and correctness of the entry procedure (see table 5.2). For further results and histograms on this questionnaire see Figs. 5.8, 5.9.

The questionnaire feedback was backed by answers from the free text section. Seven participants reported that the input process takes too long, two of which believed they could digress or even forget the PIN that they are entering. Five users reported that they confounded the numbers 6 and 9 one or several times. Two stated that they would rather use a pattern in which they do not have to search for the respective number first, i. e. a pattern of which they could easier build a mental model of or have already built it (like the numeric pad of an ATM). One participant mentioned that while focusing on the current digit, she was already searching for the next one. Three users criticized that objects in the peripheral field of view were difficult to focus on due to blurriness. Furthermore, two participants found it frustrating when many PIN codes were recognized falsely by the software, one of which wished for an undo function. To assess if there is a relationship, a Pearson correlation was run, confirming a medium positive correlation between TLX score and selection time ($r(24) = .388, p = .05$) but none between errors and TLX rating (see table 5.2). Normality was confirmed by a Shapiro-Wilk test for selection time, number of errors and TLX rating ($p > .05$ for each).

Moving Scene: Like in the abstract scenario, participants sometimes had to slow down or even briefly hold still to perform a selection, which was expressed by two participants and could be confirmed by the experimenter. Two users mentioned a noticeable drop in selection performance if the target object was in the peripheral field whereby blurriness complicated the selection. One person reported increasing eye dryness towards the end of the trial, which was due to the heat that is generated by headset and trackers. Two users said, the optical appearance and being able to "blow things up" was more appealing and enjoyable than plain cubes. Is was also mentioned that the more isolated an object is, the easier it was to select. Also, one experimentee explained that larger objects were easier to fixate, but perceived selection time was shorter for smaller objects. Again, it was both expressed by participants and observable in the course of the game that people

	Easiness		**Correctness**		**Perspicuity**		**Av. TLX**	
	Sig.	ρ	Sig.	ρ	Sig.	ρ	Sig.	r
No. of wrong PIN codes	<.001	-.74	<.001	-.836	<.05	-.461	–	.308
Average time per PIN	<.01	-.629	<.001	-.689	<.05	-.483	.05	.388

Table 5.2: **Use Case Scenario: Stationary Scene**. Using Spearman's ρ, negative correlations were found for both the total number of wrong PIN codes per participant and the time that it took to enter four digits. Ratings concerning easiness, correctness and perspicuity (UEQ) were significantly lower the more PIN codes were detected wrong or the longer it took to enter them. A Pearson test revealed a medium positive correlation between selection time and TLX score.

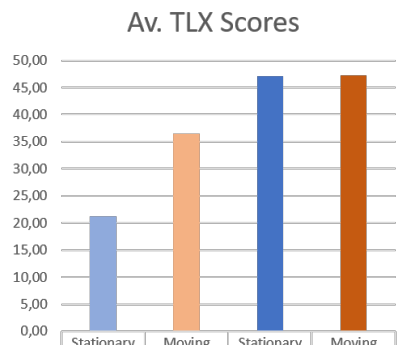
got more confident over time and did not have to search for the indicator on the floor. Several participants mentioned that is was rather disturbing if an object left the field of view and that it further complicated selection.

After the game participants were presented a form consisting of six questions (see Fig. 7.4). Eight people approved that target selection was (rather) easy, whereas 14 thought that it was (rather) not. The same tendency could be observed regarding the perceived correctness: six people agreed that their selection was interpreted correctly and 12 felt that it was not. Concerning the time it lasted until a selection was executed, a majority of 19 participants thought that it took (much) too long. When asked for the size of objects users thought have been easier to select, 19 voted in favor of large objects, whereas opinions on velocity were inconclusive. For results and histograms on this questionnaire see Fig. 5.10.

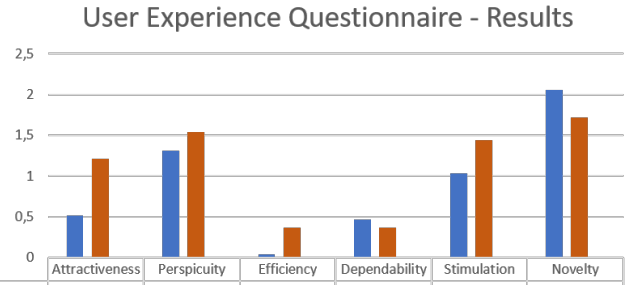
After transforming and projecting scales, mean UEQ ratings show positive tendencies in the categories attractiveness (1.21), perspicuity (1.54), stimulation (1.44) and novelty (1.72) and neutral ratings in terms of efficiency (0.37) and dependability (0.37). Again, Spearman rank-order coefficients were calculated to identify a coherence between measured performance values and subjective ratings. It confirmed a strong positive correlation between measured correctness in percent and perceived correctness as stated in the questionnaire ($r_s(24) = .572, p < .005$). Furthermore, there was a strong negative correlation found between average selection time and perceived stimulation as indicated by the User Experience Questionnaire ($r_s(24) = -.53, p < .05$). No coherence was found between measured performance values and the average TLX score by running a Pearson correlation ($r_{\text{time}}(24) = .046, r_{\text{correct}}(24) = .18$). Normality was confirmed by a Shapiro-Wilk test for selection time, correctness and TLX rating ($p > .05$ for each).

5.2.4 Discussion

The main study could partly confirm the assumptions from section 5.2 and additionally led to further insights that constitute valuable input for employing Pursuits in virtual reality. It could be validated that detection performance significantly drops when the user is moving. Analysis of the questionnaire and NASA TLX scores showed that users experienced a lot more exhaustion performing Pursuits selections while walking. Quantitative data confirmed a significant drop in selection correctness and a significant increase of selection time, which largely approves assumption one. Watching video data from when participants performed the walking task, revealed that vibration at each step led to shaky images, depending on the pace and way of walking of the individual. This led to a drop in confidence values and thereby to a reduction of gaze data that can be used for Pursuits correlation. Additionally, coordinates that were used in the calculation were distorted, which complicated and prolonged the comparison of two trajectories and led to false results. Observing users getting more confident over time, suggests that pursuing objects while navigating is learnable and gets easier after spatial positions have been internalized.



	Stationary	Moving	Stationary	Moving
Mean	21,18	36,43	47,13	47,27
SD	10,01	17,33	18,45	17,05



	Attractiveness	Perspicuity	Efficiency	Dependability	Stimulation	Novelty
Stationary	0,51	1,31	0,04	0,46	1,03	2,06
SD	1,06	0,82	1,20	1,17	0,86	0,57
Confidence	0,41	0,31	0,46	0,45	0,33	0,22
Moving	1,21	1,54	0,37	0,37	1,44	1,72
SD	0,85	0,76	0,99	1,07	0,91	0,79
Confidence	0,33	0,29	0,38	0,41	0,35	0,30

Figure 5.11: **Results from NASA TLX and User Experience Questionnaires** measured after the respective Scenario. Ratings of the UEQ were transformed and flipped to match a 7 point scale from -3 (negative) to 3 (positive) before being averaged in respect to their category. Ratings from -0.8 to 0.8 are generally considered neutral and ratings > 0.8 are considered positive for a category.

Figure 5.12: A participant performing the walking task. Due to the limited adaption range of the harness, this participant had to wear a beanie throughout the course of the study.

Throughout the main study it was obvious that performance of the implemented system varied from person to person. Therefore, differences in quantitative data as well as qualitative feedback are not completely related to the system but also due to other factors like previous experience and confidence with virtual reality, the physiognomy of the participant's head and limitations of the hardware adapting to it. Two participants had to wear a beanie as the Vive's harness was too lose, to fit their heads (see Fig. 5.12). In both cases fixation detection showed relatively poor performance, which led to long detection times during the walking task and therefore may have further promoted frustration.

According to data analysis, an increasing object size did neither result in a significant prolongation of selection time (see table 7.1) nor in loss of detection accuracy (see table 7.2). Effects on time in the stationary setting are negligibly small: coefficient $B_{\text{size.65}} = -.032$ indicates a slight decrease in time and $B_{\text{size.9}} = .006$ an even smaller increase. While walking, only the largest size prolongs selection time by 0.15s compared to the reference situation ($p = .615$). Also correctness is not significantly influenced by object size: the largest effect again is observable in the walking scenario: when increasing size from the smallest to the largest value, the probability of a correct selection drops by 4.8% ($p = .406$). A tendency can be seen that a larger size decreases selec-

tion performance, especially when walking. As none of these effects are significant, the second assumption could not be confirmed.

The third assumption suggested that a higher radius, i.e. objects being in the outer field of view, would result in poorer performance while walking. Results show that a change of trajectory radius while sitting did not lead to a significant change in selection time. Coefficients show a very small decrease of selection time with an increasing radius, but this change is not significant ($p_{\text{radius1.5}} = .985$, $p_{\text{radius2.5}} = .946$). Also, the effect on correctness is very scarce. In contrast, if the participant is moving, coefficients indicate an effect that is totally opposed to the assumption: an increasing radius significantly decreases detection time and increases selection correctness. The explanation of this fact can have several origins: all five objects on the trajectory had the same angular velocity ($45°s^{-1}$). Equation 6 describes the coherence between trajectory velocity v and angular velocity ω if $\vec{v}$ if orthogonal to $\vec{\omega}$. A decrease in radius therefore results in a decrease in trajectory velocity. Earlier in this section, the effect of walking on gaze data was described. A concussion that leads to the headset shaking has a much larger impact if the eyes are moving slowly when pursuing an object of smaller velocity than on an object with a larger velocity. When being presented a constellation with the smallest radius, some participants mentioned that they had the impression the objects were not moving at all. Obviously, when the user is navigating, world-bound objects are moving quickly through the field of view, which can cause slow, user-bound objects appear motionless. If an object is not moving, smooth pursuits do not occur and the eye starts alternating between fixations and saccades [31]. Thus the algorithm cannot perform.

$$v = \omega r \tag{6}$$

The fact that all selections in the abstract scenario that were performed below 900 milliseconds (i.e. in the second correlation cycle) were false, is due to the design of the system, more specifically to the correlation window that averages coefficients within the given time window of 900 ms. As the highest possible value from the Pearson correlation is 1 and the default value for averaging the coefficients is zero at the beginning, the highest possible result is 0.5 after the first cycle, which is below the pre-set Pearson threshold of 0.6. That is why no selection has been performed after the first cycle. So for a selection in the second cycle, there had to be two consecutive correlations with a result of over 0.6. The latency until a saccade occurs after a stimuli was presented, ranges from 120 to 350 ms. In addition to that comes the time of the saccade itself, which "is roughly 20 - 30 ms plus 2 ms for every degree of amplitude" [49]. This means that data gathered in the first time frame is completely unrelated to any object movement and a resulting high coefficient is purely accidental. It is for these random peaks that the correlation window was introduced, and its functionality is visible after the 5th or 6th cycle, where the percentage of successful recognitions stays relatively constant.

The increase of TLX values from stationary to mobile setting in the abstract scenario is clearly ascribable to the walking task, as it was the only changed variable. The two scenes of the use case scenario are not comparable regarding TLX scores due do the distinctness of the two tasks. The fact that task load in the stationary scene was detected to be almost as high as in the mobile scene although the participant did not have to perform the walking task, leads to the conclusion that the PIN input procedure itself is very demanding, if not too demanding. Feedback from the participants clearly indicates that the duration for entering a digit takes too long. This prolonged mental workload, the need to first search for an object, then fixate on it for an average $4.63s$, combined with the heat coming from the headset and eye trackers, understandably led to fatigue and frustration, especially if the input ended up being wrong. To increase input speed, the selection process could be altered either by presenting less numbers at once or providing more distinct trajectories. Also, searching for the number should be avoided to give the user the opportunity to build a mental model of where the input elements are. UEQ results reflect this very well: the interaction received low ratings for efficiency and attractiveness, but still received high or very

high ratings for categories stimulation and novelty (see Fig. 5.11). However, a majority of 13 participants would use this kind of authentication at least once a month as it seems to be perceived secure (see Fig. 5.8).

The asteroid shooting game received better ratings for attractiveness, although, efficiency and dependability ratings were low and it was perceived as rather difficult and lengthy to select an object. In contrast to the PIN entry, participants got immediate feedback about their selection and the task was more entertaining, given the added sound and special effects. This was the only scene in which trajectories could be partly outside the field of view, which was not perceived well and rather irritating. During the session, the experimenter adjusted the counter threshold individually to find a compromise between selection time and correctness. Counter values were adjusted to the participants during the session: participants whose eyes could only poorly be tracked needed a higher threshold, resulting in longer selection times. Thus, the threshold value at the end of a session represented the individual tracking quality. Large differences were ascertained: Final thresholds ranged from 12 to 23, and the resulting mean selection times per participant have a range of $\Delta_{\text{mean}} = 13.46s$. Longer selection times and erroneous selections led to high TLX ratings and poor usability ratings. This shows that there is a need for improvement of the adaptability of the trackers.

6 Conclusion

In this thesis, a system was developed that employs the Pursuits approach for selecting objects by gaze in virtual reality. Pursuits detects a fixation on a moving object by calculating Pearson correlations for gaze and object coordinates over time, rather than using the absolute gaze position. The system was developed with an HTC Vive headset equipped with binocular eye trackers. Additional to the Pearson threshold th that correlation coefficients have to exceed to be considered for a selection and correlation window w that defines the length of the time window, over which coordinates are correlated, the system features further parameters to ensure correctness: a confidence threshold that ensures sufficient quality of the gaze data received from the eye-trackers, a time window w_c within which the last calculated coefficients are averaged to filter out sudden peaks of the Pearson coefficient and an optional correlation counter, that counts the number of Pearson coefficients that were larger than th, which can be compared to a counter threshold to determine a selection. Additionally, the frequency of the correlation procedure f can be adjusted to either increase reactivity to reduce CPU load, and . The implementation is described in section 4.3. The following settings were found credible and used throughout the study: $w = 300ms, th = 0.6, f = 0.3, w_c = 900ms$.

The performance of the system was evaluated in a user study, which involved two parts: first, a stimuli-free abstract environment to investigate the effects of *trajectory radius*, *object size*, *perceived depth* and *user movement* on selection time and selection correctness for the selection of one out of five objects moving on a circular trajectory. Second, two use case scenarios in which interacting with objects in VR with pursuits was demonstrated. In the first scenario, users had to authenticate while standing in front of an ATM machine by entering a PIN using gaze. In the second, users could blast bypassing asteroids with their gaze. The study procedure is described in section 5.2.

Due to individual physiognomy of the participants, distance between eyes and screen varied and for this reason, objects of the same size occupied a changing amount of space in the users' field of view. A measurement method to determine a participant's field of view and to convert units used in the development software (Unity 3D) to degrees of visual angle was developed and is described in section 4.4. This method is used to determine the values in table 5.1 of which mean values are used to report results concerning trajectory radius and object size in this section.

In the abstract scenario, the first coefficient that matched the criteria ($th = 0.6, w_c = 900ms$) was used to perform selections of which on average 75% were correct while the participant was sitting, and 57% while the participant was walking. Given that five objects were visible at the same time, a rate of 20% would represent chance. In the use case scenarios, coefficients greater than th were counted for each object over time. The first object to exceed a counter threshold was selected. This led to a correct detection of 83% of digits in a PIN entry scenario, in which users were standing, and 75% correct targets in an asteroid shooting game where users were walking.

Analysis of the abstract part led to the result that of the four independent variables "movement" and "radius" have a significant effect on detection performance. None of the object parameters had a significant effect on detection performance when the user was *not moving*. However, there are some tendencies: an object size of 17.14° hat a worsening effect on correctness compared to a size of 12.22°. Also, increasing the distance to the user from 2 to 16 meters led to a slight drop in correctness. Selection performance deteriorated significantly, when users were *moving*: mean selection time increased by a factor of 2.52 and mean correctness dropped by 18%. Under this premise, increasing trajectory radius significantly improved performance: a raise from 4.84° to 22.94° resulted in an increase of 16.6% for a correct selection and a decrease of 1.77 seconds for the estimated duration. This is due to three reasons: first, walking as a main task leads to an increase in workload as two visual tasks have to be performed at the same time. Second, slow objects, i.e. objects with a small radius, appear motionless if the environment is moving. A motionless target is not pursued by the eyes, which complicates correlation. Third, vibrations caused by walking impact tracking performance due to the harness material and the weight of the

headset. This has a larger effect when following slower objects so in a way, a large, fast trajectory compensates for a shaking headset.

Qualitative evaluation of the use cases revealed that entering a PIN using gaze instead of the traditional number pad was not perceived well. As this procedure requires correct input the counter threshold was set to a high number (20), which led to longer input durations of on average 21.4 seconds for one PIN. This is way too long for a procedure performed with gaze, especially if there is no feedback about correctness, and leads to high physical demand. Shooting objects with the eyes was generally perceived as more attractive than solely performing selections due to the feedback received immediately after the activity.

In summary, the following conclusions for Pursuits in VR can be derived from the information gathered in trial and main studies for user-bound objects during this project: circular trajectories are in general perceived to be more convenient to follow. In a setting in which the user remains in the same position, any combination of size, radius and depth can be utilized. It is suggested to not exceed an object size of 17.14°, as selection correctness showed a decreasing tendency. Long processes should be subdivided into several steps between which the user's eyes can rest and feedback is given about the success of the performed entry. If Pursuits is to be utilized while the user is moving, a poorer detection performance is to be expected. An object radius of 14.25° or larger helps compensate vibrations that occur due to walking and prevents eyes from performing fixations instead of smooth pursuits. If the scene is not familiar yet, users tend to be overwhelmed when having to navigate and perform pursuit selections at the same time. In both scenarios that involved walking, performance improved after users became familiar with the scene. Thus, a scene in which the main task involves navigating to random targets will probably result in poor performance. Object trajectories should not exceed the user's field of view and should be avoided in areas where objects could become blurry. If correctness is crucial, the implemented coefficient counter has proven to improve correctness and to a certain degree compensate for poor tracking.

7 Future Work

Insights were gathered with a specific hardware setting, i.e. an HTC Vive equipped with the binocular Vive Add-On distributed by Pupil Labs. There were outcomes specific to this setting, like blurriness at the edges of the field of view, which may have had an impact on detecting objects being in this specific area. Results may differ when utilizing a different hardware setting, e.g. an Oculus Rift. As the whole software was developed on the Unity platform, which also provides first-party support for the Oculus, this study should be easily repeatable without further amendments.

Position and sharpness of the user's eyes are a crucial factor for reliable eye-tracking. The vibration caused by walking was the reason why the user's eyes did not stay in a constant position in the camera image. A better attachment mechanism of the headset could at least contain these vibrations and help improve tracking. Also, image stabilization as it is implemented in current digital cameras would help filtering out impacts caused by movement. The quality of tracking data also relies on the eyes being in focus. As the cameras' focus can only be adjusted manually the headset has to be put on and off the head every time a focus adjustment is made, which is a tedious, inaccurate procedure. Due to the short distance, it occurred that even if pupils were in focus when looking straight, they might get blurry if the eyes move to fixate on an object at the edge of the field of view, which is reason number two. With better tracking data, a higher Pearson threshold could have been set, which itself would have enabled to set a lower counter threshold in the use case scenarios. The introduction of the fixation counters antagonized false positives and led to a more reliable selection, however, at the cost of time. If future generations of eye-trackers for virtual reality headsets include a possibility to manipulate camera focus from outside the headset or even feature an auto-focus function, detection performance could significantly increase and therefore a shorter correlation procedure would be possible.

The reported sizes in visual angles were gathered using measurement data from the procedure described in section 4.4. This procedure relies on the participant correctly reading the presented scale and reporting object positions properly as it is impossible to confirm this information. So these values might slightly differ from the real sizes. The performance of the system varied strongly between participants, even if images were sharp. It would be interesting to determine, if the physiognomy of a person, like interpupillary distance and the measured field of view, correlate with tracking performance.

A neutral point shown in the center of the view when no stimuli are present would have ensured that all participants started the search for the visual target from the same position and take the same time to fixate on the target object. Then saccade times could have been calculated according to table 5.1 and for radii {0.5,1.5,2.5} saccade times would have resulted in {39-49,77-87,112-122} ms [49]. Although these time spans are very short, this amendment would have increased consistency in the procedure and should be taken into account for future investigations.

The variable "depth" was considered a factor for the study used out of curiosity to ascertain if it has some sort of effect. A deteriorating tendency of increasing depth on detection correctness was found. Also while walking, according to the generalized linear model, longest selection times are to be expected with a medium depth. There were no additional objects given in the stationary scene of the abstract scenario that would have served as a depth clue. In the walking scene, only play area bounds were visible, as participants needed to at least see the floor while walking. Although none of these effects are significant, this would be an interesting factor to investigate in future studies.

The results of the study suggest that there is a positive effect of an increasing radius at the time the user is moving. As the absolute object speed on a trajectory is directly proportional to the angular speed ω a larger radius also means greater absolute speed. A study investigating absolute speed and radius separately would be a worthwhile supplement to the findings of this thesis.

As calibrated data is used by the proposed system to perform correlations, occasional recalibration is needed. Subject for a further thesis on Pursuits in virtual reality could be the employment of uncalibrated raw data, as it is provided by the Pupil Labs tracking software. This includes the conversion of camera coordinates into user bound coordinates of the virtual scene considering both cameras' exact tilt and rotation.

In this section it has become obvious that there are some issues and questions left to be investigated, to make Pursuits for VR more reliable and faster. This thesis is seen as a first of many steps towards enabling truly spontaneous gaze interaction in virtual reality.

DVD Contents

- **DVD 1** Unity Main Study Folder Without the "Assets" folder, which is on DVD 2. The Assets folder has to be integrated into the main study folder. Then the project can be opened in Unity. The folder also contains the logfiles of the main study:
  - `AbstractStationary`: Scene no. 1
  - `AbstractWalking`: Scene no. 2
  - `UseCaseStationary`: Scene no. 3
  - `PursuitsShooter`: Scene no. 4
- **DVD 2**
  - `zu DVD 1`: contains the Assets folder for the Unity project on disc 1
  - `Matlab Tobii Integration`: contains the first attempts to enable Eye Tracking with a Tobii Rex in MatLab
  - `PreStudy`: Unity Project Folder of the Trial Study
  - `Video Recordings`: recordings taken during the main study
  - `Latex Files`: folder containing this Latex project
  - `Auswertung`: containing SPSS and Excel files and WebReports of the data analyses

Literature

[5] Scott W Greenwald et al. “Technology and Applications for Collaborative Learning in Virtual Reality”. In: *12th International Conference on Computer Supported Collaborative Learning*. Philadelphia, PA: International Society of the Learning Sciences, 2017. DOI: 10.22318/cscl2017.115.

[6] Kate Laver et al. “Virtual reality for stroke rehabilitation”. In: *Stroke* 43.2 (2012), pp. 20–22. ISSN: 00392499. DOI: 10.1161/STROKEAHA.111.642439.

[10] Anjul Patney et al. “Towards foveated rendering for gaze-tracked virtual reality”. In: *ACM Transactions on Graphics* 35.6 (2016), pp. 1–12. ISSN: 07300301. DOI: 10.1145/2980179.2980246. URL: http://dl.acm.org/citation.cfm?doid=2980179.2980246.

[11] Päivi Majaranta and Andreas Bulling. “Eye Tracking and Eye-Based Human-Computer Interaction”. In: 2014, pp. 39–65. ISBN: 978-1-4471-6391-6. DOI: 10.1007/978-1-4471-6392-3_3. URL: http://link.springer.com/10.1007/978-1-4471-6392-3%7B%5C_%7D3.

[12] Darren Gergle, Robert E Kraut, and Susan R Fussell. “Using Visual Information for Grounding and Awareness in Collaborative Tasks”. In: *Human-Computer Interaction* 28.June 2013 (2013), pp. 1–39. ISSN: 07370024. DOI: 10.1080/07370024.2012.678246. URL: http://www.tandfonline.com/doi/abs/10.1080/07370024.2012.678246.

[13] Susan E. Brennan et al. “Coordinating cognition: The costs and benefits of shared gaze during collaborative search”. In: *Cognition* 106.3 (Mar. 2008), pp. 1465–1477. ISSN: 00100277. DOI: 10.1016/j.cognition.2007.05.012. URL: http://linkinghub.elsevier.com/retrieve/pii/S0010027707001448.

[14] Margaret M. Bradley et al. “The pupil as a measure of emotional arousal and autonomic activation”. In: *Psychophysiology* 45.4 (2008), pp. 602–607. ISSN: 1469-8986. DOI: 10.1111/j.1469-8986.2008.00654.x. URL: http://dx.doi.org/10.1111/j.1469-8986.2008.00654.x.

[15] Joseph H Goldberg and Anna M Wichansky. “Eye Tracking in Usability Evaluation: A Practitioner’s Guide”. In: *The Mind’s Eye - Cognitive and Applied Aspects of Eye Movement Research*. Ed. by Ralph Radach, Jukka Hyona, and Heiner Deubel. 1st ed. Oxford: Elsevier Science, 2002. Chap. 23, pp. 493–517. ISBN: 9780444510204. URL: https://www.elsevier.com/books/the-minds-eye/radach/978-0-444-51020-4.

[16] Mohamed Khamis, Florian Alt, and Andreas Bulling. “A Field Study on Spontaneous Gaze-based Interaction with a Public Display Using Pursuits”. In: *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers* (2015), pp. 863–872. DOI: 10.1145/2800835.2804335. URL: http://doi.acm.org/10.1145/2800835.2804335.

[17] Jörg Müller et al. “Requirements and Design Space for Interactive Public Displays”. In: *Proceedings of the 18th ACM International Conference on Multimedia*. MM ’10. Firenze, Italy: ACM, 2010, pp. 1285–1294. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1874203. URL: http://doi.acm.org/10.1145/1873951.1874203.

[18] Mélodie Vidal, Andreas Bulling, and Hans Gellersen. “Pursuits”. In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp ’13*. New York, New York, USA: ACM Press, 2013, p. 439. ISBN: 9781450317702. DOI: 10.1145/2493432.2493477. URL: http://dl.acm.org/citation.cfm?doid=2493432.2493477%7B%5C%%7D5Cnhttp://dl.acm.org/citation.cfm?id=2493477%20http://dl.acm.org/citation.cfm?doid=2493432.2493477.

[19] Hamilton Hartridge and LC Thomson. “Methods of investigating eye movements”. In: *The British journal of ophthalmology* 32.9 (1948), p. 581.

[20] Norman H Mackworth and Edward Llewellyn Thomas. “Head-mounted eye-marker camera”. In: *JOSA* 52.6 (1962), pp. 713–716.

[21] Tom N Cornsweet and Hewitt D Crane. “Accurate two-dimensional eye tracker using first and fourth Purkinje images”. In: *JOSA* 63.8 (1973), pp. 921–928.

[22] RJ Jacob and Keith S Karn. “Eye tracking in human-computer interaction and usability research: Ready to deliver the promises”. In: *Mind* 2.3 (2003), p. 4.

[23] John Merchant, Richard Morrissette, and James L Porterfield. “Remote measurement of eye direction allowing subject motion over one cubic foot of space”. In: *IEEE Transactions on Biomedical Engineering* 4 (1974), pp. 309–317.

[24] James L Levine. “Performance of an eyetracker for office use”. In: *Computers in biology and medicine* 14.1 (1984), pp. 7781–7989. DOI: 10.1016/0010-4825(84)90022-2.

[25] Carlos H. Morimoto and Marcio R M Mimica. “Eye gaze tracking techniques for interactive applications”. In: *Computer Vision and Image Understanding* 98.1 (2005), pp. 4–24. DOI: 10.1016/j.cviu.2004.07.010.

[26] Z Zhu and Q Ji. “Eye gaze tracking under natural head movements”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005* I (2005), pp. 918–923. DOI: 10.1109/CVPR.2005.148. URL: http://ieeexplore.ieee.org/document/1467364/%7B%5C%%7D0Ahttp://www.scopus.com/inward/record.url?eid=2-s2.0-33745151109%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=.

[27] Yusuke Sugano and Andreas Bulling. “Self-Calibrating Head-Mounted Eye Trackers Using Egocentric Visual Saliency”. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST ’15*. New York, New York, USA: ACM Press, 2015, pp. 363–372. ISBN: 9781450337793. DOI: 10.1145/2807442.2807445. URL: http://dl.acm.org/citation.cfm?doid=2807442.2807445.

[28] Pieter Blignaut et al. “Improving the Accuracy of Video-Based Eye Tracking in Real Time through Post-Calibration Regression”. In: *Current Trends in Eye Tracking Research*. Cham: Springer International Publishing, 2014, pp. 77–100. ISBN: 9783319028682. DOI: 10.1007/978-3-319-02868-2_5. URL: http://link.springer.com/10.1007/978-3-319-02868-2%7B%5C_%7D5.

[29] Ken Pfeuffer et al. “Pursuit calibration”. In: *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST ’13*. New York, New York, USA: ACM Press, 2013, pp. 261–270. ISBN: 9781450322683. DOI: 10.1145/2501988.2501998. URL: http://dl.acm.org/citation.cfm?id=2501998%20http://dl.acm.org/citation.cfm?doid=2501988.2501998.

[30] Wolfgang Fuhl et al. “Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art”. In: *Machine Vision and Applications* (June 2016). ISSN: 0932-8092. DOI: 10.1007/s00138-016-0776-4. URL: http://link.springer.com/10.1007/s00138-016-0776-4.

[31] Robert J K Jacob. “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 Empowering people - CHI ’90*. 1990, pp. 11–18. ISBN: 0201509326. DOI: 10.1145/97243.97246. URL: http://portal.acm.org/citation.cfm?doid=97243.97246.

[32] Kari-Jouko Majaranta, Päivi and Räihä. “Twenty Years of Eye Typing : Systems and Design Issues Twenty Years of Eye Typing : Systems and Design Issues”. In: *Proceedings of Eye Tracking Research and Applications* 1.212 (2002), pp. 15–22. DOI: 10.1145/507072.507076.

[33] Stephen Vickers, Howell Istance, and Aulikki Hyrskykari. “Performing Locomotion Tasks in Immersive Computer Games with an Adapted Eye-Tracking Interface”. In: *ACM Transactions on Accessible Computing (TACCESS)* 5.1 (2013), p. 2. ISSN: 1936-7228. DOI: 10.1145/2514856. URL: http://dl.acm.org/citation.cfm?id=2531922.2514856.

[34] Andrew T. Duchowski. “A breadth-first survey of eye-tracking applications”. In: *Behavior Research Methods, Instruments, & Computers* 34.4 (Nov. 2002), pp. 455–470. DOI: 10.3758/BF03195475. URL: http://www.springerlink.com/index/10.3758/BF03195475.

[35] Keith Rayner. “Eye movements in reading and information processing: 20 years of research.” In: *Psychological bulletin* 124.3 (1998), p. 372. DOI: dx.doi.org/10.1037/0033-2909.124.3.372.

[36] Andrew Schall. *Eye Tracking in User Experience Design*. 2014, pp. 3–26. ISBN: 9780124081383. DOI: 10.1016/B978-0-12-408138-3.00001-7. URL: http://www.sciencedirect.com/science/article/pii/B9780124081383000017.

[37] Joseph H Goldberg and Xerxes P Kotval. “Computer interface evaluation using eye movements: methods and constructs”. In: *International Journal of Industrial Ergonomics* 24.6 (Oct. 1999), pp. 631–645. ISSN: 01698141. DOI: 10.1016/S0169-8141(98)00068-7. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169814198000687.

[38] Alex Poole and Linden J. Ball. “Eye Tracking in HCI and Usability Research”. In: *Encyclopedia of Human Computer Interaction* (2006), pp. 211–219. DOI: 10.4018/978-1-59140-562-7.ch034. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59140-562-7.ch034.

[39] Keith Rayner et al. “Integrating text and pictorial information: Eye movements when looking at print advertisements.” In: *Journal of Experimental Psychology: Applied* 7.3 (2001), pp. 219–226. DOI: 10.1037//1076-898X.7.3.219.

[40] Pradipta Biswas and Pat Langdon. “A new input system for disabled users involving eye gaze tracker and scanning interface”. In: *Journal of Assistive Technologies* 5.2 (2011), pp. 58–66. DOI: 10.1108/17549451111149269.

[41] Veronica Sundstedt. “Gazing at Games: Using Eye Tracking to Control Virtual Characters”. In: *ACM SIGGRAPH 2010 Courses* (2010), 5:1–5:160. DOI: 10.1145/1837101.1837106. URL: http://dl.acm.org/citation.cfm?id=1837106%7B%5C%%7D5Cnhttp://doi.acm.org/10.1145/1837101.1837106.

[42] Jayson Turner et al. “EyePlay: Applications for Gaze in Games”. In: *Proceedings of the First ACM SIGCHI Annual Symposium on Computer-human Interaction in Play* (2014), pp. 465–468. DOI: 10.1145/2658537.2659016. URL: http://doi.acm.org/10.1145/2658537.2659016.

[43] Eduardo Velloso et al. “Arcade+: A Platform for Public Deployment and Evaluation of Multi-Modal Games”. In: *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. CHI PLAY ’15. London, United Kingdom: ACM, 2015, pp. 271–275. ISBN: 978-1-4503-3466-2. DOI: 10.1145/2793107.2793145. URL: http://doi.acm.org/10.1145/2793107.2793145.

[44] Mohamed Khamis et al. “GazeTouchPass: Multimodal Authentication Using Gaze and Touch on Mobile Devices”. In: *Proceedings of the 34th Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’16. San Jose, CA, USA: ACM, 2016. DOI: 10.1145/2851581.2892314.

[45] Andreas Bulling, Florian Alt, and Albrecht Schmidt. “Increasing the security of gaze-based cued-recall graphical passwords using saliency masks”. In: *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI ’12* (2012), p. 3011. DOI: 10.1145/2207676.2208712. URL: http://dx.doi.org/10.1145/2207676.2208712.

[46] Dh Cymek et al. “Entering PIN Codes by Smooth Pursuit Eye Movements”. In: *Jemr* 7.4 (2014), pp. 1–11. ISSN: 19958692. DOI: 10.16910/jemr.7.4.1.

[47] Mingxin Yu et al. “Human-Robot Interaction Based on Gaze Gestures for the Drone Teleoperation”. In: *Journal on eye moment research* 7.4 (2014), pp. 1–14. DOI: 10.16910/jemr.7.4.4. URL: https://www.researchgate.net/profile/Mingxin%7B%5C_%7DYu2/publication/273945849%7B%5C_%7DHuman-Robot%7B%5C_%7DInteraction%7B%5C_%7DBased%7B%5C_%7Don%7B%5C_%7DGaze%7B%5C_%7DGestures%7B%5C_%7Dfor%7B%5C_%7Dthe%7B%5C_%7DDrone%7B%5C_%7DTeleoperation/links/5578346008aeacff20021c16.pdf.

[48] Yanxia Zhang et al. “Eye tracking for public displays in the wild”. In: *Personal and Ubiquitous Computing* 19.5-6 (Aug. 2015), pp. 967–981. ISSN: 1617-4909. DOI: 10.1007/s00779-015-0866-8. URL: http://link.springer.com/10.1007/s00779-015-0866-8.

[49] Roger HS Carpenter. *Movements of the Eyes, 2nd Rev*. Pion Limited, 1988. ISBN: 0850861098.

[50] Augusto Esteves et al. “Orbits”. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST ’15*. 2015. ISBN: 9781450337793. DOI: 10.1145/2807442.2807499.

[51] S. Stellmach and R. Dachselt. “Designing gaze-based user interfaces for steering in virtual environments”. In: *Proc. of ETRA’12*. Vol. 1. 212. New York, New York, USA: ACM Press, 2012, pp. 131–138. ISBN: 978-1-4503-1221-9. DOI: 10.1145/2168556.2168577. URL: http://dl.acm.org/citation.cfm?doid=2168556.2168577.

[52] Vildan Tanriverdi and Robert J. K. Jacob. “Interacting with Eye Movements in Virtual Environments”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’00. The Hague, The Netherlands: ACM, 2000, pp. 265–272. ISBN: 1-58113-216-6. DOI: 10.1145/332040.332443. URL: http://doi.acm.org.emedien.ub.uni-muenchen.de/10.1145/332040.332443.

[53] Felix Hülsmann, Timo Dankert, and Thies Pfeiffer. “Comparing gaze-based and manual interaction in a fast-paced gaming task in Virtual Reality”. In: *Proceedings of the Workshop Virtuelle & Erweiterte Realität 2011*. Ed. by Christian-A. Bohn and Sina Mostafawy. Shaker Verlag, 2011, pp. 1–12. URL: https://pub.uni-bielefeld.de/publication/2308550.

[56] Anjul Patney et al. “Perceptually-based Foveated Virtual Reality”. In: *ACM SIGGRAPH 2016 Emerging Technologies*. SIGGRAPH ’16. Anaheim, California: ACM, 2016, 17:1–17:2. ISBN: 978-1-4503-4372-5. DOI: 10.1145/2929464.2929472. URL: http://doi.acm.org/10.1145/2929464.2929472.

[59] Moritz Kassner, William Patera, and Andreas Bulling. “Pupil: An Open Source Platform for Pervasive Eye Tracking and Mobile Gaze-based Interaction”. In: *CoRR* abs/1405.0006 (2014). URL: http://arxiv.org/abs/1405.0006.

[60] E. Bruce Goldstein. *Sensation and Perception*. Ed. by Mariane Tafliner et al. 6th ed. Pacific Grove, CA, USA: Wadsworth, 2002, pp. 227–233. ISBN: 0534539645.

[61] Sandra G. Hart. “Nasa-Task Load Index (NASA-TLX); 20 Years Later”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9 (2006), pp. 904–908. DOI: 10.1177/154193120605000909. eprint: http://dx.doi.org/10.1177/154193120605000909. URL: http://dx.doi.org/10.1177/154193120605000909.

[64] B Laugwitz et al. “Subjektive Benutzerzufriedenheit quantitativ erfassen: Erfahrungen mit dem User Experience Questionnaire UEQ”. In: *Usability Professionals* (2009), pp. 220–225. URL: https://www.researchgate.net/profile/Martin%7B%5C_%7DSchrepp/publication/236462743%7B%5C_%7DSubjektive%7B%5C_%7DBenutzerzufriedenheit%7B%5C_%7Dquantitativ%7B%5C_%7Derfassen%7B%5C_%7DErfahrungen%7B%5C_%7Dmit%7B%5C_%7Ddem%7B%5C_%7DUser%7B%5C_%7DExperience%7B%5C_%7DQuestionnaire%7B%5C_%7DUEQ/links/00b7d52457033316ab000000.pdf.

Web-References

[1] Deloitte. *Virtual Reality treibt Entwicklung der Unterhaltungselektronik an*. URL: https://www2.deloitte.com/de/de/pages/presse/contents/virtual-reality-treibt-unterhaltungselektronik-an.html (visited on July 4, 2017).

[2] Thiemo Heeg. *So cool ist Virtual Reality schon jetzt*. URL: http://www.faz.net/aktuell/wirtschaft/netzwirtschaft/vr-brillen-virtual-reality-soll-das-naechste-grosse-ding-werden-14414639.html (visited on July 4, 2017).

[3] Tristan Parrish Moore. *What Steam's Data Reveals About The Health Of VR's Ecosystem*. May 2017. URL: https://www.vrfocus.com/2016/07/what-steams-data-reveals-about-the-health-of-vrs-ecosystem/ (visited on July 4, 2017).

[4] Ben Lang. Feb. 2017. URL: https://www.roadtovr.com/valve-confirms-development-on-three-full-vr-games-not-experiments/ (visited on July 4, 2017).

[7] Peter Ilg. Jan. 2017. URL: http://www.zeit.de/mobilitaet/2017-01/audi-virtual-reality-autohaus-vertrieb-verkaufsprozess (visited on July 4, 2017).

[8] Sophie Charara. Aug. 2015. URL: https://www.wareable.com/vr/british-museum-samsung-gear-vr-headset-party-667 (visited on July 4, 2017).

[9] Uwe Gerritz. Jan. 2017. URL: http://hessenschau.de/kultur/vr-technik-im-museum-virtuelle-dinos-sind-publikumshit,virtuelle-welten-im-museum-100.html (visited on July 4, 2017).

[54] Tobii Gaming. *Enhanced Avatar Interactions with Eye Tracking in VR Games*. Mar. 2017. URL: https://medium.com/@tobiigaming/enhanced-avatar-interactions-with-eye-tracking-in-vr-games-1bc6886fcfb3 (visited on July 4, 2017).

[55] Ben Lang. July 2017. URL: https://www.roadtovr.com/nvidia%7B-%7D%7B-%7Dbased-foveated-rendering-research/ (visited on July 4, 2017).

[57] Sean Buckley. *This Is How Valve's Amazing Lighthouse Tracking Technology Works*. May 19, 2015. URL: http://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768 (visited on June 9, 2017).

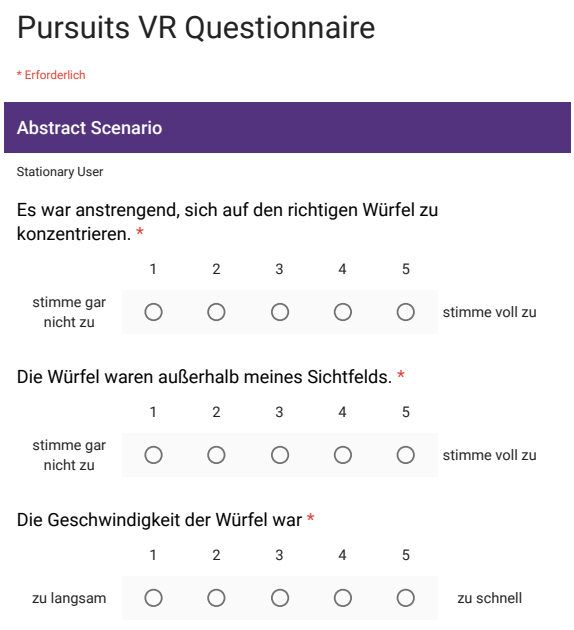
[58] JP Hawkins. *Valve Brings SteamVR to the Unity Platform*. Feb. 10, 2016. URL: https://blogs.unity3d.com/2016/02/10/valve-brings-steamvr-to-the-unity-technologies-platform/ (visited on June 9, 2017).

[62] Jens Grubert. *NASA Task Load Index German Short*. URL: http://jensgrubert.bplaced.net/nasa-tlx-short/TLX-German-short.html (visited on May 29, 2017).

[63] Jens Grubert. *NASA Task Load Index English Short*. URL: http://jensgrubert.bplaced.net/nasa-tlx-short/TLX-English-short.html (visited on May 29, 2017).

[65] Andreas Hinderks, Martin Schrepp, and Jörg Thomaschewski. *UEQ - User Experience Questionnaire*. URL: http://www.ueq-online.org (visited on May 29, 2017).

Appendix I: Forms


Pursuits VR Questionnaire

* Erforderlich

Abstract Scenario

Stationary User

Es war anstrengend, sich auf den richtigen Würfel zu konzentrieren. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

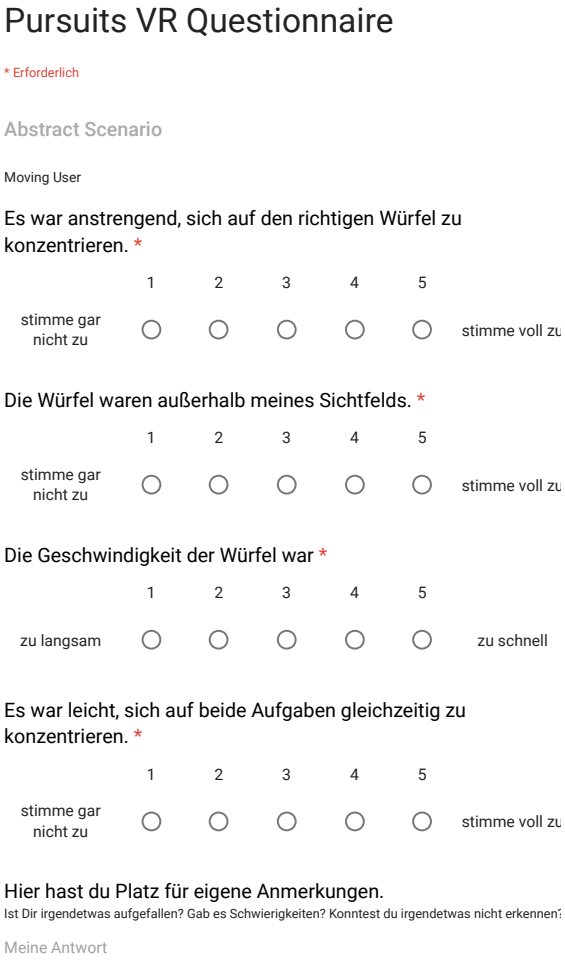
Die Würfel waren außerhalb meines Sichtfelds. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Die Geschwindigkeit der Würfel war *

	1	2	3	4	5	
zu langsam	○	○	○	○	○	zu schnell

Figure 7.1: Questionnaire shown to participants after the first scene.


Pursuits VR Questionnaire

* Erforderlich

Abstract Scenario

Moving User

Es war anstrengend, sich auf den richtigen Würfel zu konzentrieren. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Die Würfel waren außerhalb meines Sichtfelds. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Die Geschwindigkeit der Würfel war *

	1	2	3	4	5	
zu langsam	○	○	○	○	○	zu schnell

Es war leicht, sich auf beide Aufgaben gleichzeitig zu konzentrieren. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Hier hast du Platz für eigene Anmerkungen.

Ist Dir irgendetwas aufgefallen? Gab es Schwierigkeiten? Konntest du irgendetwas nicht erkennen?

Meine Antwort

Figure 7.2: Questionnaire shown to participants after the first scene.

Pursuits VR Questionnaire

* Erforderlich

Use Case Scenario

Stationary User

Es war einfach, mit der vorgestellten Methode eine PIN einzugeben. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Ich finde die vorgestellte Eingabemethode sicher. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Ich würde die Eingabemethode im Alltag nutzen. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Wie häufig wären Sie bereit, die vorgestellte Eingabemethode zu nutzen? *

- ○ Nie
- ○ einmal pro Monat
- ○ einmal am Tag (z.B. Eingabe der PIN am Geldautomaten)
- ○ mehrmals am Tag (z.B. Smartphone entsperren)

Das System hat meine Eingaben richtig erkannt. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Die Eingabe hat zu lange gedauert. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Die Eingabe war anstrengend für meine Augen. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Ich habe mich vom Geldautomaten weggedreht, um die Auswahlen zu treffen. *

	1	2	3	4	5	
nie	○	○	○	○	○	immer

Hier hast Du Platz für eigene Anmerkungen.
Was hat Dir gefallen? Was hat Dir nicht gefallen? Was würdest Du verbessern?

Meine Antwort

Figure 7.3: Questionnaire shown to participants after the third scene.

Pursuits VR Questionnaire

* Erforderlich

Use Case Scenario

Moving User

Die Objekte waren einfach auszuwählen *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Das System hat meine Auswahl richtig erkannt. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Es war einfach, die beiden Aufgaben gleichzeitig auszuführen. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Die Auswahl hat zu lange gedauert. *

	1	2	3	4	5	
stimme gar nicht zu	○	○	○	○	○	stimme voll zu

Die Auswahl war einfacher bei *

	1	2	3	4	5	
langsamen Objekten	○	○	○	○	○	schnellen Objekten

Die Auswahl war einfacher bei *

	1	2	3	4	5	
kleinen Objekten	○	○	○	○	○	großen Objekten

Hier hast Du Platz für deine eigenen Anmerkungen.
Was hat Dir gefallen? Was hat Dir nicht gefallen? Was würdest Du verbessern?

Meine Antwort

Figure 7.4: Questionnaire shown to participants after the fourth scene.

Pursuits VR Questionnaire

* Erforderlich

User Experience Questionnaire

Stationary User

Der nachfolgende Fragebogen besteht aus Gegensatzpaaren von Eigenschaften, die das Produkt haben kann. Abstufungen zwischen den Gegensätzen sind durch Kreise dargestellt. Durch Ankreuzen eines dieser Kreise kannst Du Deine Zustimmung zu einem Begriff äußern.

Entscheide möglichst spontan. Es ist wichtig, nicht lange über die Begriffe nachzudenken, damit Deine unmittelbare Einschätzung zum Tragen kommt.

Bitte kreuze immer eine Antwort an,

auch wenn Sie bei der Einschätzung zu einem Begriffspaar unsicher sind oder finden, dass es nicht so gut zum Produkt passt. Es gibt keine „richtige" oder „falsche" Antwort!

*

	1	2	3	4	5	6	7	
unerfreulich	○	○	○	○	○	○	○	erfreulich

*

	1	2	3	4	5	6	7	
unverständlich	○	○	○	○	○	○	○	verständlich

*

	1	2	3	4	5	6	7	
kreativ	○	○	○	○	○	○	○	phantasielos

*

	1	2	3	4	5	6	7	
leicht zu lernen	○	○	○	○	○	○	○	schwer zu lernen

*

	1	2	3	4	5	6	7	
wertvoll	○	○	○	○	○	○	○	minderwertig

*

	1	2	3	4	5	6	7	
langweilig	○	○	○	○	○	○	○	spannend

*

	1	2	3	4	5	6	7	
uninteressant	○	○	○	○	○	○	○	interessant

*

	1	2	3	4	5	6	7	
un-berechenbar	○	○	○	○	○	○	○	voraussagbar

*

	1	2	3	4	5	6	7	
schnell	○	○	○	○	○	○	○	langsam

*

	1	2	3	4	5	6	7	
originell	○	○	○	○	○	○	○	konventionell

*

	1	2	3	4	5	6	7	
behindernd	○	○	○	○	○	○	○	unterstützend

*

	1	2	3	4	5	6	7	
gut	○	○	○	○	○	○	○	schlecht

*

	1	2	3	4	5	6	7	
kompliziert	○	○	○	○	○	○	○	einfach

*

	1	2	3	4	5	6	7	
abstoßend	○	○	○	○	○	○	○	anziehend

*

	1	2	3	4	5	6	7	
herkömmlich	○	○	○	○	○	○	○	neuartig

*

	1	2	3	4	5	6	7	
unangenehm	○	○	○	○	○	○	○	angenehm

*

	1	2	3	4	5	6	7	
sicher	○	○	○	○	○	○	○	unsicher

*

	1	2	3	4	5	6	7	
aktivierend	○	○	○	○	○	○	○	einschläfernd

*

	1	2	3	4	5	6	7	
erwartungs-konform	○	○	○	○	○	○	○	nicht erwartungs-konform

Figure 7.5: User Experience Questionnaire Part I

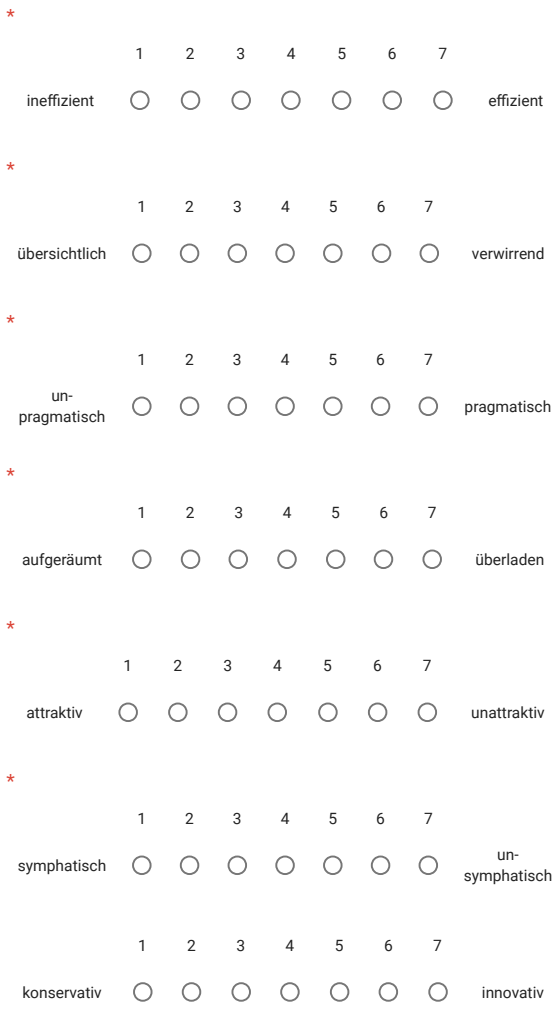


Figure 7.6: User Experience Questionnaire Part II

Appendix II: Figures and Tables

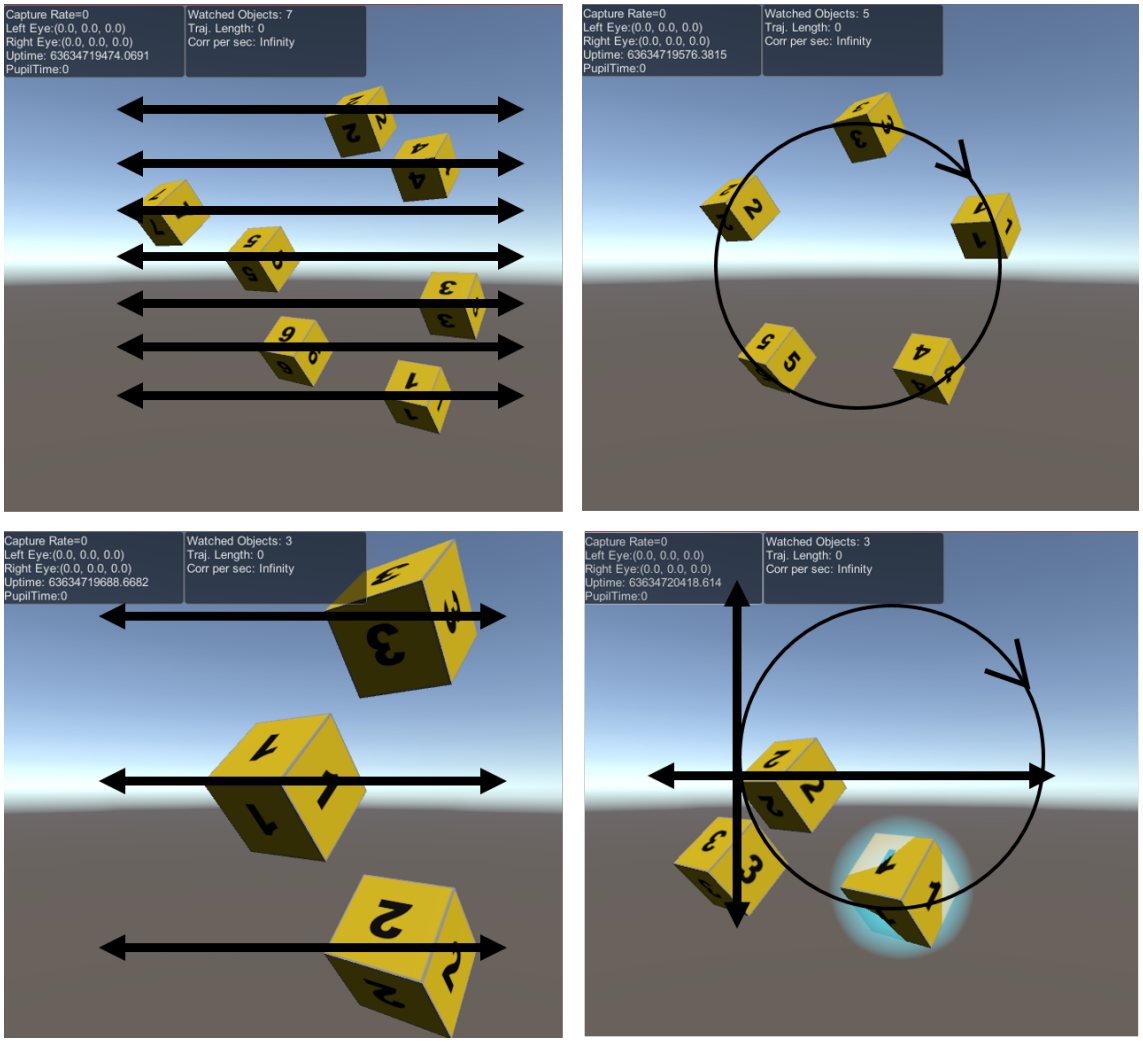


Figure 7.7: Scene views of the trial study. Black arrows were inserted for depicting movement directions. *Top left:* Scene 1a. In this scene the maximum objects on vertical and horizontal trajectories was sounded out. *Top right:* Scene 1b. In this scene the maximum number of objects on a circular trajectory was tested. *Bottom left:* Scene 3. Scene for testing different object sizes. *Bottom right:* Scene for comparing horizontal, vertical and circular trajectories side by side. Cube 1 is highlighted.

Figure 7.8: Scene views of all settings in the main study. Boxes in the top of the images are only visible to the experimenter. *Top Left:* Abstract scenario, stationary setting. Cubes rotating in a stimuli-free black ennvironment. Users are told to fixate on the red cube. *Top Right:* Use case scenario, moving setting. Outer space setting with asteroids moving in the field of view. Direction indicator is in front of the user. One asteroid is about to explode. *Bottom left:* Use case scenario, stationary setting. ATM machine screen providing feedback about progress of entry process. Ten cubes are rotating in the field of view and fixating on a certain number enters the respective digit. *Bottom right:* Abstract Scenario, moving scene. Only cubes and area bounds are visible.

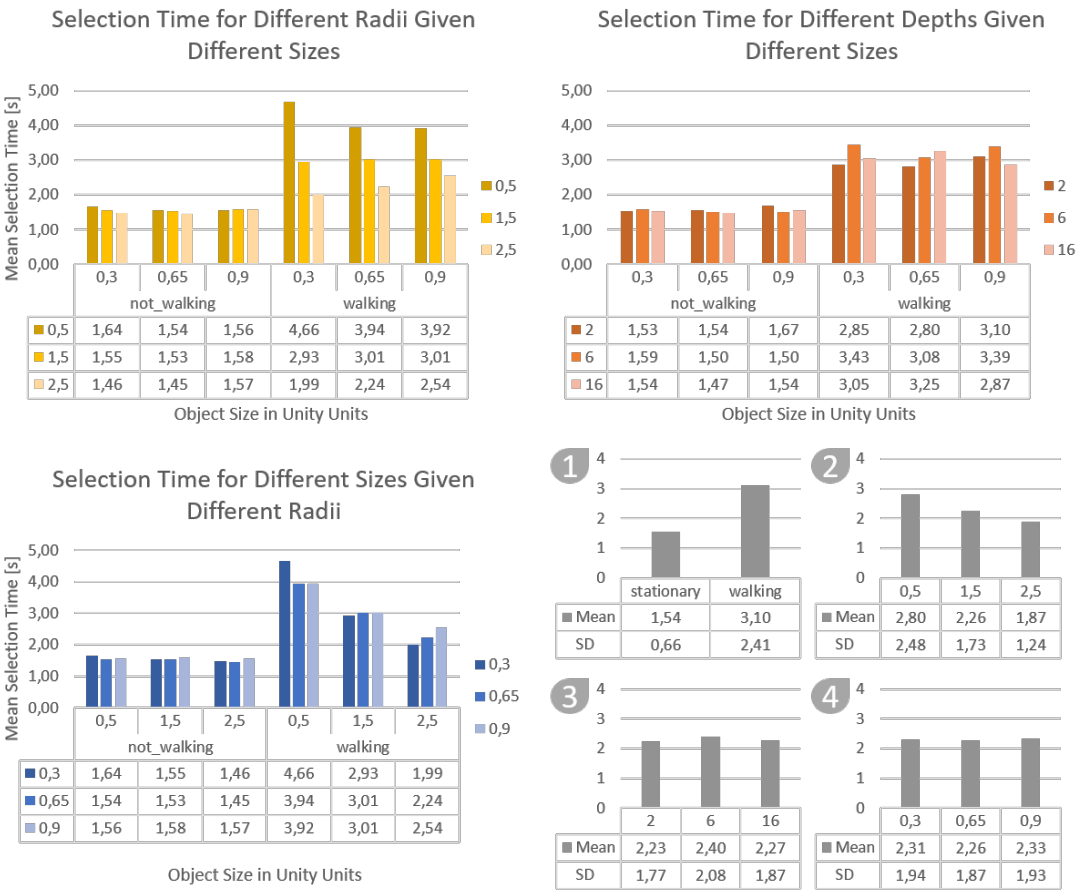


Figure 7.9: Graphs depicting mean selection times for the abstract scenario of the main study. Large graphs in the top row and bottom left show mean values of stationary and walking settings. *Top Left*: Combination of size (category) and **radius** (color). *Top Right*: Combination of size (category) and **depth** (color). *Bottom Left*: Combination of radius (category) and **size** (color). *Bottom Right*: Four graphs depicting the mean values for every independent variable on its own. (1) Movement, (2) Radius, (3) Depth, (4) Size.

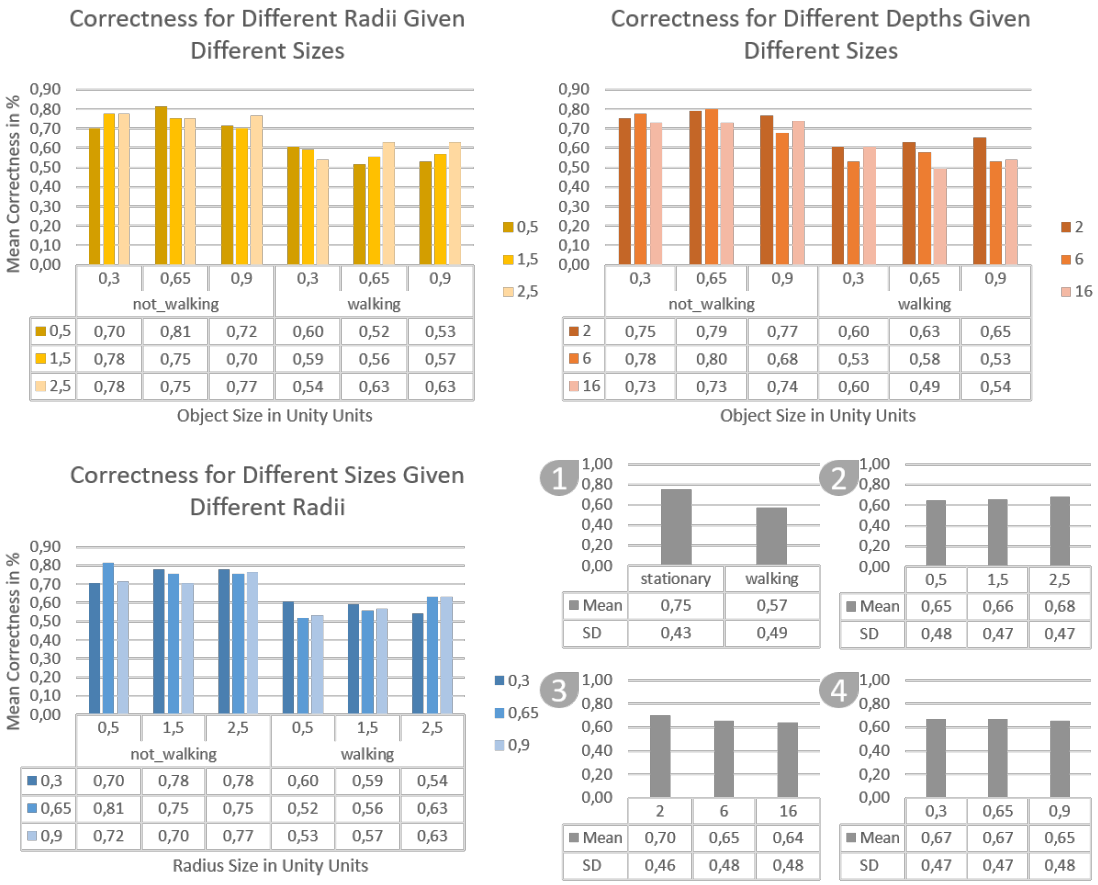


Figure 7.10: Graphs depicting mean correctness for the abstract scenario of the main study. Large graphs in the top row and bottom left show mean values of stationary and walking settings. *Top Left*: Combination of size (category) and **radius** (color). *Top Right*: Combination of size (category) and **depth** (color). *Bottom Left*: Combination of radius (category) and **size** (color). *Bottom Right*: Four graphs depicting the mean values for every independent variable on its own. (1) Movement, (2) Radius, (3) Depth, (4) Size.

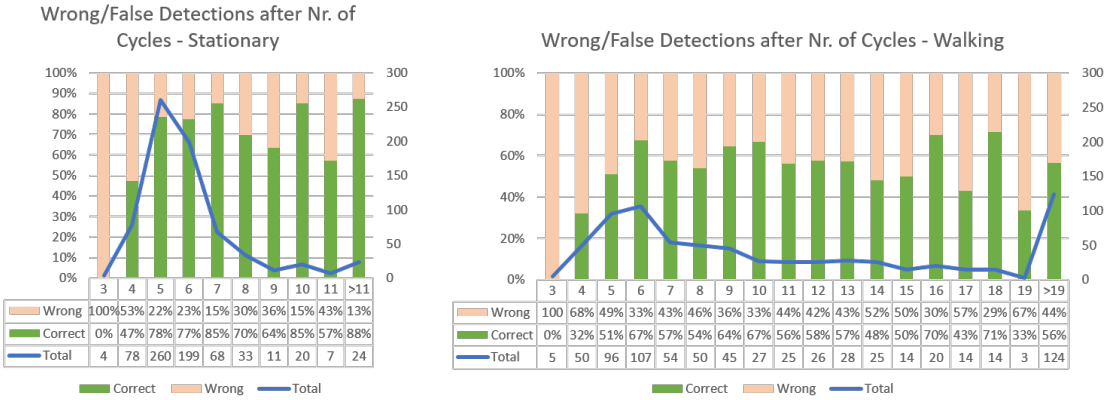


Figure 7.11: Ratio of correct and wrong detections for every correlation cycle. The left graph shows values for the stationary setting, in which most recognitions occured in a time frame below 3 seconds. The right graph shows correct/false ratios for the walking scenario. In this setting, selection times were generally longer so selections are distributed over more cycles. The x-axis represents the number of cycles. One cycle duration is 0.3 s.

Variables	*Regression Coefficient B*	*Std. Error*	*Sig.*	*Exp(B)*	est. time [s]
radius1.5	-.016	.046	.735	.985	1.60
radius2.5	-.055	.046	.229	.946	1.54
size0.65	-.032	.046	.485	.968	1.58
size0.9	.006	.046	.902	1.006	1.64
depth6	-.030	.046	.517	.971	1.58
depth16	-.043	.046	.355	.958	1.56
walking=yes	.875	.0718	.000	2.4	3.90
radius1.5*walking=yes	-.318	.0661	.000	.727	2.80
radius2.5*walking=yes	-.550	.0662	.000	.577	2.13
size=0.65*walking=yes	.022	.0657	.733	1.023	3.86
size=0.9*walking=yes	.033	.0659	.615	1.034	4.05
depth=6*walking=yes	.115	.0657	.08	1.122	4.25
depth=16*walking=yes	.092	.0659	.164	1.096	4.10
Constant	.487	.0497	.000	1.627	1.63

Table 7.1: **Results of Generalized Linear Model** for observed detection correctness values. The smallest values for each category have been set as reference situation (i.e. radius=0.5, depth=2, size=0.3, walking=no). Values in the last column are expected times according to the model and in combination with the reference parameters set for the categories not mentioned in the first column.

Variables	*Regression Coefficient B*	*Std. Error*	*Sig.*	*Exp(B)*	P_{correct}
radius1.5	.048	.218	.827	1.049	78.9%
radius2.5	.000	.217	1	1	78.1%
size0.65	.099	.223	.656	1.104	79.7%
size0.9	-.184	.215	.392	.832	74.7%
depth6	-.049	.221	.825	.952	77.2%
depth16	-.189	.217	.386	.828	74.6%
walking=yes	-1.369	.312	.000	.254	47.5%
radius1.5*walking=yes	.524	.288	.069	1.689	61.6%
radius2.5*walking=yes	.678	.287	.018	1.970	64.1%
size=0.65*walking=yes	-.046	.291	.847	.955	48.8%
size=0.9*walking=yes	.237	.285	.406	1.267	42.7%
depth=6*walking=yes	-.147	.291	.614	.864	42.7 %
depth=16*walking=yes	-.199	.288	.488	.819	38.0%
Constant	1.269	.238	.000	3.558	78.1%

Table 7.2: **Binomial logistic regression results** for observed detection correctness values. The smallest values for each category have been set as reference situation (i.e. radius=0.5, depth=2, size=0.3, walking=no). Values in the last column are expected probabilities according to the model and in combination with the reference parameters set for the categories not mentioned in the first column.