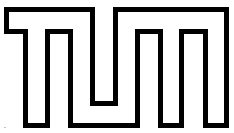


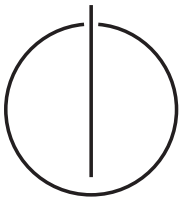
Technische Universität München

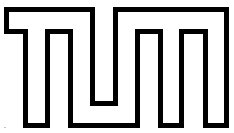
Fakultät für Informatik

Bachelor Thesis in Informatik: Games Engineering

A study of the effect of hand posture in 3D pointing

Nikolas Schneider





Technische Universität München

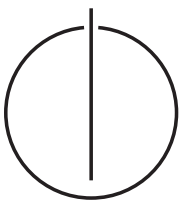
Fakultät für Informatik

Bachelor Thesis in Informatik: Games Engineering

A study of the effect of hand posture in 3D pointing

Eine Studie über den Effekt von Handstellungen für Deuten in 3D

Author: Nikolas Schneider
Supervisor: Prof. Gudrun Klinker
Advisor: Frieder Pankratz. Ph.D.
Date: April 15th, 2016



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, April 15th, 2016 Nikolas Schneider

Abstract

On a desktop computer several user interfaces and macros support users who are completing a variety of different tasks, whereas robots and virtual agents do not always support real time input through button pushes. Using a head-mounted display (*HMD*) to show a user interface with a variety of functions, similar to *WIMP* (*Windows Icon Menu Pointer*) interfaces, would clutter the user's field of vision. Therefore, there is motivation to develop more natural methods of issuing commands. This thesis explores leveraging the expressivity of the hands for this. The advantage of hand postures is that they allow a natural way to communicate or to accomplish everyday tasks. Defining accurate and intuitive commands which can be performed by gesture and posture control is a challenge, since hand postures, unlike button presses, are fluid and interconnected. Additionally, findings from various motor control literature suggest that not every posture is suitable for pointing [11]. This thesis attempts to identify the effects of posture on pointing accuracy and to quantify the performance differences of two postures compared to the standard pointing posture. Furthermore, the thesis also shows results from a controlled user study in which subjects point at several targets. These suggest that the posture which was most "distant" from the standard pointing posture performed worse, but also that it was the posture that showed the biggest learning potential [11].

Zusammenfassung

Auf einem Computer werden Benutzer von verschiedenen Interfaces und Makros beim Bewältigen von verschiedenen Aufgaben unterstützt, während Roboter und virtuelle Agenten nicht immer eine Eingabe durch das Drücken von Knöpfen in Echtzeit ermöglichen. Die Verwendung von einem Head-Mounted Display (*HMD*) zum Anzeigen von Benutzerinterfaces mit sämtlichen Basisfunktionen und ähnlichem Aufbau wie *WIMP* (*Windows Icon Menu Pointer*) würde die Sicht stören. Somit gibt es einen Beweggrund um natürlichere Methoden zum Erteilen von Befehlen zu entwickeln, indem die Expressivität der Hände dafür verwendet wird. Der Vorteil dabei ist, dass sich diese für den Menschen natürlich anfühlen, da Menschen ihre Hände stets verwenden, sowohl bei der Kommunikation untereinander, als auch beim Bewältigen von alltäglichen Aufgaben. Genaue und intuitive Befehle zu definieren, die durch Gestenkontrolle und Körperhaltungskontrolle ausgeführt werden können, ist eine Herausforderung, da Handgesten, im Gegensatz zum Drücken von Knöpfen, fließend ineinander übergehen. Erkenntnisse aus der Fachliteratur im Bereich der motorischen Steuerung suggerieren, dass sich nicht jede Handstellung zum Deuten eignet [11]. Diese Arbeit verdeutlicht die Auswirkungen der Genauigkeit von Gesten und vergleicht die Performance der Gesten im Vergleich zu dem gewöhnlichen Deuten. Im Zuge dieser Arbeit wurde weiterhin eine Studie durchgeführt, in der die Probanden mehrere Ziele durch das Deuten auf diese treffen mussten, um die verschiedenen Handstellungen in den Bereichen Performanz und Genauigkeit zu testen [11].

Contents

1 Introduction

Today in the field of human-robot interaction there are new ways of interacting with virtual reality, besides the conventional methods of using a desktop computer. Some of these use gestures, postures, facial expressions and interaction through commands with the human voice. These alternatives do not go beyond the possibilities of a desktop computer, as they do not possess the complexity of a keyboard; simplified but distinct commands are needed to reach a given goal. One possibility would be using voice recognition to give commands to a virtual agent. However, a certain degree of ambient silence is required for reliable recognition, therefore the use of human voice is not suitable for a working environment. In addition, voice commands can be ambiguous and the resulting noise could be distracting to other people in the vicinity.
Simplifying commands through gesture and posture control could be a possibility to avoid those kinds of issues, especially postures are both time efficient and executable without making a lot of noise. Furthermore, they are more intuitive than the deployment of a desktop computer in augmented reality but also simplifies the execution of technical knowledge in comparison to the traditional way of human to machine interaction. Since gestures might not work well, and require a larger scale of space to perform, hand postures were used in this study [8]. Another positive aspect of using the hand is that a high variety of commands can be performed by using different postures. When regarding various hand postures, it has to be taken into account that, on one hand, some postures are more or less efficient than the standard pointing posture, especially in terms of pointing and selecting in a 3D environment. On the other hand, different postures could provide an opportunity to diversify commands bound to certain postures.
Therefore, the goal of this thesis is to compare the effectiveness of different hand postures for pointing in a 3D environment in terms of accuracy and performance, and to discuss possible fields of use. This was achieved with a user study.
In the following sections related work, a depiction of the used postures, a description of the user study, an evaluation of the data, conclusion and future work will be introduced.

2 Related Work

While some researchers have explored different ways to issue remote commands Nancel et al. used the off-hand to augment the pointing to ultra-high resolution wall-sized displays [6] [9] [8]. The authors made use of different hand-held devices, in addition to the dominant hand which was using different ray-casting techniques to select, point, and zoom into specific areas of the wall-sized display (Figure: 2.1) [5] [4]. Their work shows that gestures had the lowest performance and were less efficient than device based techniques. Also, gesture based input techniques are vulnerable to fatigue when performed over a longer period of time. Considering these results, this study investigates the performance of postures instead of gestures. Additionally, this study utilizes a hand tracking device and passive IR retroreflective markers to track the hand position in a three dimensional environment.
Lopes et. al made use of the proprioceptive sense of users, to allow eyes-free interaction for wearables [7]. The users interacted with an implemented wearable device by forming a pose with their wrist. The device was not able to read or find the position of limbs. In addition to that, people do not like to wear attachments on their arm. This approach uses the human body as an input device, similar to the study.
Like another relevant paper, this paper evaluates different postures in a user study by considering the accuracy and comfort of the posture that was used [11]. Different hand postures instead of arm postures were used. Because of the similarity of this paper to previous studies, a decreased efficiency is expected when an uncomfortable or unnatural posture is used.
Another approach features hand recognition through a low resolution camera, like a web cam, by defining a small on-screen workspace [3]. A system was implemented whereby users were able to use their index finger to navigate the mouse cursor on a desktop PC and complete simple tasks like clicking and scrolling. An accurate usage of the index finger instead of a mouse could not be achieved, due to the limited video quality a web cam is able to provide, and the differences of its resolution and the desktop screen. Therefore, this paper does not support any tasks a mouse could perform, but instead, similar to the approach of Gope et al., it supports hand tracking with the goal to point at targets and to find a way to point accurately in a 3D, instead of a 2D, environment [3]. In addition to that, the system could be extended in a way, that through an UI on a *HMD* a system similar to *WIMP* in AR environment could be realized, or to take aim at different positions and to give orders to an agent or robot without any other UI support.

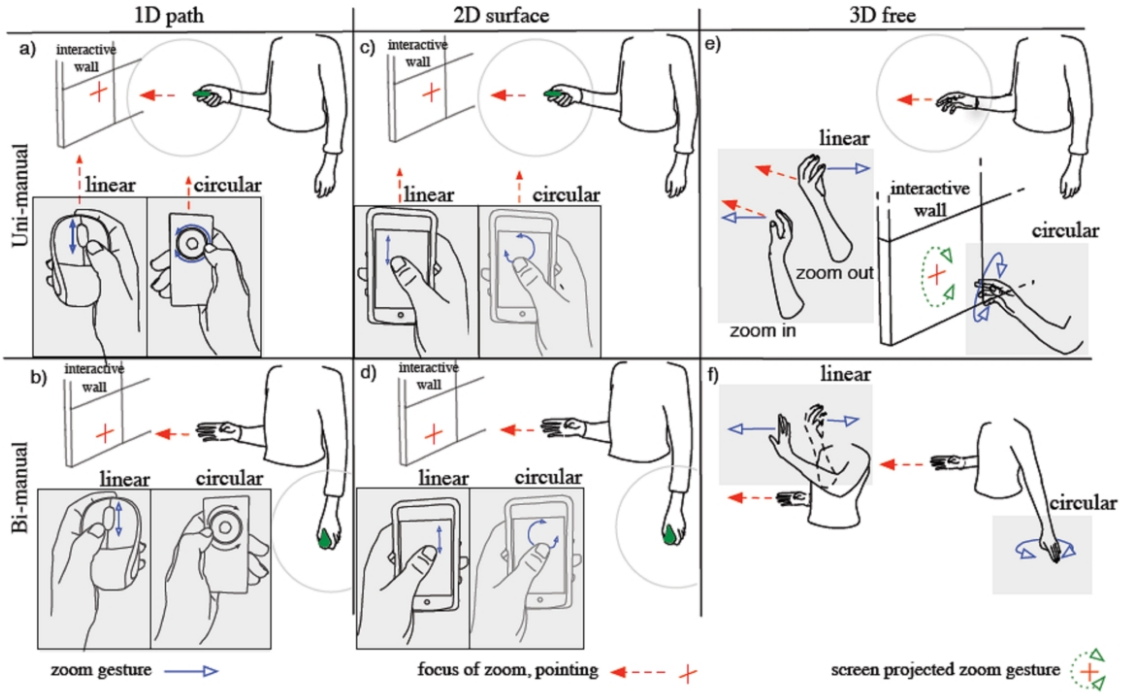


Figure 2.1: Techniques used to augment pointing [8]

3 Posture Control

This section provides an overview of postures and why they were chosen, compared to the standard pointing posture that was used. In this study, a total of four postures were selected and then three of them were compared in terms of performance, accuracy, and comfort.

3.1 Standard Pointing Posture

The most important posture and main subject of comparison is the *Pointing* posture, in the following defined as standard pointing posture (Figure: 3.1). It was selected to provide an already familiar and comfortable way to point in a 3D environment and therefore, grant a basis of comparison for the other two postures.

3.2 Spiderman Posture

The *Spiderman* posture was selected due to its resemblance to the standard pointing posture (Figure: 3.1). In contrast to *Pointing*, the *Spiderman* posture is more stressful and less intuitive, when it comes to showing in a direction, but provides a natural "cross-hair" between the index and the little finger, which could result in a higher accuracy. It was also selected due to its potential of using the tilt of the hand and the finger orientation for more than just pointing tasks in the future.

Figure 3.1: Standard pointing posture (left), *Spiderman* posture (right)

3.3 Okay Posture

This posture was chosen to maximize the stress on the hand and to observe, how this effects performance and accuracy compared to the standard posture (Figure: 3.2). Out of every posture, the *Okay* posture was the one that was expected to be the least instinctive, but it was also believed to be accurate, because it bore a likeness to writing and pointing with a pen.

3.4 Idle State Posture

The *Idle State* posture is the only one in the setup which was not compared to the standard pointing posture (Figure: 3.2). Its main function was to allow the subjects to change the posture between two sets in a comfortable way and to avoid distracting the posture recognition algorithm, by allowing the sensors to resense the current hand position.

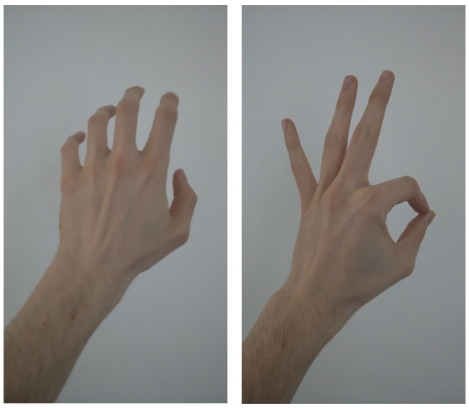


Figure 3.2: Posture for *Idle State*(left), *Okay* posture (right)

4 Evaluation

After clarifying the postures used in the project, this chapter focuses on the execution as well as the basic concepts underlying to carry out the study. It utilized a *Leap Motion Controller*, an *ART tracking system*, an *Oculus Rift*, and the *Unity 3D Engine* for tracking hand postures in a 3D environment.

4.1 Study Design

The main task of this study is to analyze the time and accuracy of a hand pointing in a three-dimensional augmented reality environment. To ensure the same starting conditions and the highest posture recognition rate, each subject calibrated an individual dataset of finger positions and then, participated in a total of three blocks, consisting of three sets of twelve trials per tested hand posture. These sets within a block of hand postures were *Pointing*, *Okay*, and *Spiderman*. The postures were selected randomly and, then, repeatedly used in the same order in the second and third block. In each of them, the subject was aiming at one of twelve targets that were ordered in a hemispherical way (Figure: 4.1) with each target spawned randomly one after another. When the subjects were satisfied with the current position they were aiming at, they had to confirm the position with a left click on the mouse. Each target is identical in size to any other target. To make sure that the subject was not able to prepare, the test only begins after a countdown.

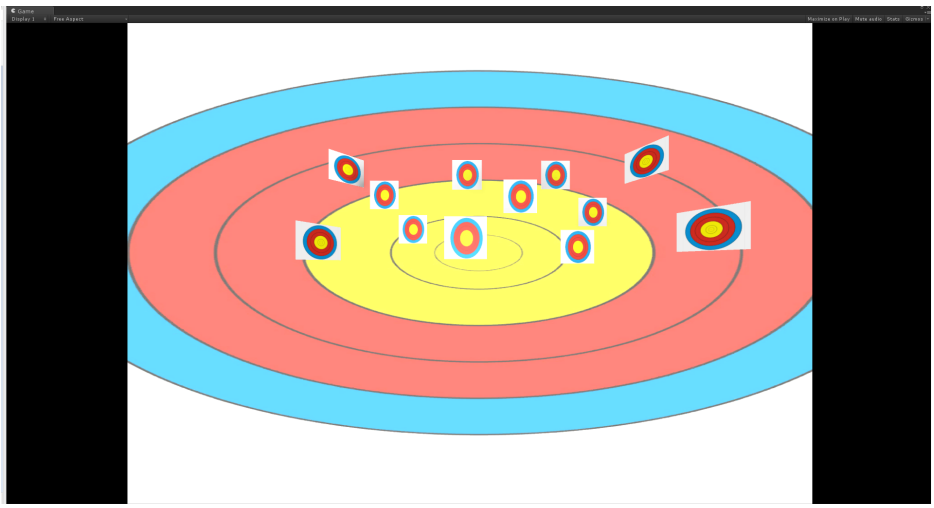


Figure 4.1: Position of all twelve targets

4.2 Visualization

To assure that the subject utilized the correct hand position and to give some guidance where exactly the subjects' hand posture was aiming at, a colored line was rendered. If the currently detected posture was not the correct one, or if it was not recognized at all, the line was colored red, otherwise it was colored green (Figure: 4.2). Additionally, the targets were not visible, as long as the posture was not correctly identified. When no set was in progress, no line was displayed.
Internally, this line was represented by a *ray*, a mathematical half-line, defined by an initial point of origin and a direction. Rays can be tested against collisions with other objects, which was used in this project to determine if a target was being hit.
Each hand posture had its own aiming technique. The *Pointing* posture defined the position and the direction of the ray solely through the index finger, which was the instinctive most natural way of pointing at an object. The *Spiderman* posture used the midpoint between the little finger and the index finger to determine the direction, while the *Okay* pose used the point where the thumb and index finger met.(Figure: 4.3)

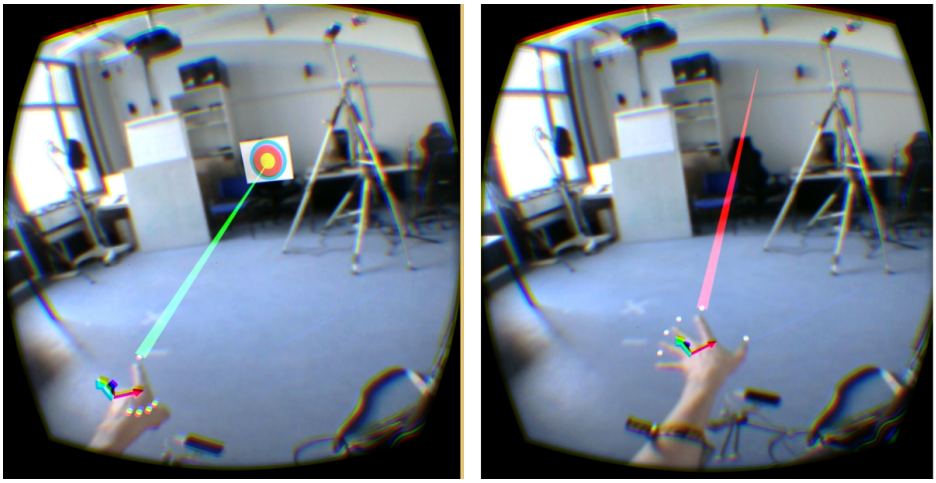


Figure 4.2: Posture recognized (left), posture not recognized due to false posture (right)

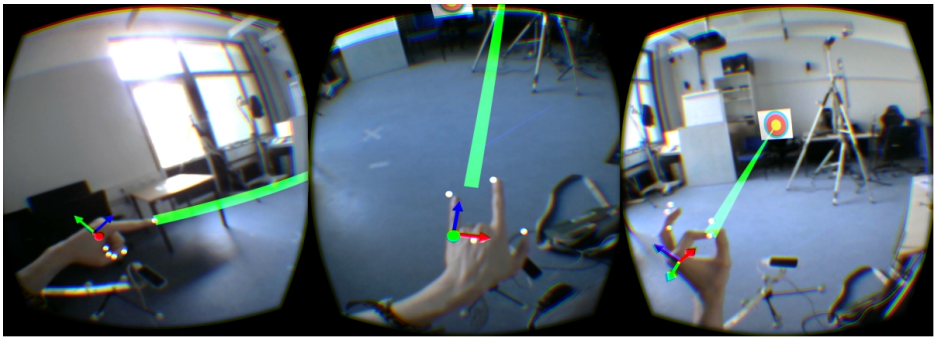


Figure 4.3: *Pointing*(left), *Spiderman* (middle), *Okay* (right)

4.3 Data Tracking

While a subject was taking the test, all relevant data was logged to a .csv file. Each time the subject confirmed the final aiming position, a line was added to the text file with the name, age, dominant-hand, gender, block number of the set, current posture mode, set number, time to target, overall time for the trial, and the distance from the position the subject was pointing at to the center of the target, in this order (Figure: 4.4). In addition to the .csv file, each subject had to answer a small set of questions to evaluate their opinion and to give a ranking of which posture was the most accurate and most intuitive, according to their personal experience.

```
subjectName,22|,right,m,1,Okay,1,5.880871,22.58691,0.09884571
subjectName,22,right,m,1,Okay,1,2.785376,25.37229,0.2979828
subjectName,22,right,m,1,Okay,1,4.504764,29.87707,0.09940409
subjectName,22,right,m,1,Okay,1,3.595921,33.47302,0.1057877
subjectName,22,right,m,1,Okay,1,14.75088,48.22393,0.03256427
subjectName,22,right,m,1,Okay,1,3.294938,51.5189,0.1566286
subjectName,22,right,m,1,Okay,1,1.447061,52.96597,0.138841
                                    ⋮
```

Figure 4.4: Example of the .csv file format used for the evaluation

4.4 Statistical Results

After the research and the evaluation of the raw user data, the following results emerged. The postures were analyzed regarding the time taken to aim, the overall time needed to complete the set of trials, and the distance to the target center, from this point forward referred to as deviation. The lower the deviation value, the more accurate and the closer to the center the subject hit. A total of 15 subjects participated in this study.
To confirm the significance of the data a repeated-measure ANOVA (*Analysis of variance*) was performed. The repeated-measure ANOVA is the extension of the dependent t-test and is designed to test related, but not independent groups in terms of significance. This kind of ANOVA requires at least one independent and one dependent variable. In this project posture, gender, and handedness were the independent variables, while the deviation and the time to aim were the dependent variables. Additionally, a post-hoc Tukey's test was performed to confirm the significance and independence among the postures.
Due to the fact that an ANOVA solely identifies significant differences within groups, a post-hoc test is used to determine significance among different groups. Since there is an equal amount of tests for each posture, a Tukey's test provides accurate results.
The repeated-measure ANOVA test found a significant effect on time to aim ($F_{(2,1617)} = 23.2$, $p < 0.001$). A Tukey's post-hoc test confirmed the statistical significance effect between the *Okay* and *Spiderman* ($p < 0.001$) and between the *Pointing* and *Okay* ($p < 0.001$) posture. There was no statistical significance between the *Pointing* and *Spiderman* posture.
Yet, a statistical significance in the overall time measurement ($F_{(2,28)}= 14{,}99$, $p < 0.001$) was proven via a repeated-measure ANOVA test and a post-hoc Tukey's test between the *Okay* and *Spiderman* ($p < 0.001$) and among the *Pointing* and *Okay* ($p < 0.001$). No significance regarding the *Pointing* and *Spiderman* was discovered ($p = 0.172$). The same procedure in the category deviance revealed that a significance with $F_{(2,1617)} = 10.893$ and $p < 0.001$ with a significant post-hoc test result between *Pointing* and *Spiderman* ($p < 0.05$) as well as between *Pointing* and *Okay* ($p < 0.001$) was achieved. Also, no statistical significance value among the *Okay* and *Spiderman* posture ($p = 0.0647$) was determined.

The ANOVA has also shown a statistical significance in both the handedness and the gender of the subject in terms of deviance, time to target, and overall time to complete a trial ($p < 0.05$).

4.5 Observation and Interpretation

To analyze the overall data, a first look at the single set results was taken (Figure: 4.5). First off, as seen in the figure, the time it took to target with *Pointing* and *Spiderman* was only divergent during the first set and was almost the same in the following. More interesting is the fact that the time it took to aim using the *Okay* posture was almost halved over the three blocks, from an average of around 6 seconds in the first iteration down to 3.1 seconds in the last, while the other two postures did not improve significantly. Therefore, it is possible to assume that the *Okay* posture either benefited from a learning effect while the other two postures did not improve after the second set or that the *Okay* pose was so unnatural, that an acclimatization was necessary in the first two sets. Due to the weight of the metal contraption attached to the subjects' forearm during the study, a fatigue effect could be

observed after the second block, increasing the time to aim (Figure: 4.6). Additionally, the fatigue effect was observed more dominantly in terms of accuracy in each block (Figure: 4.7).

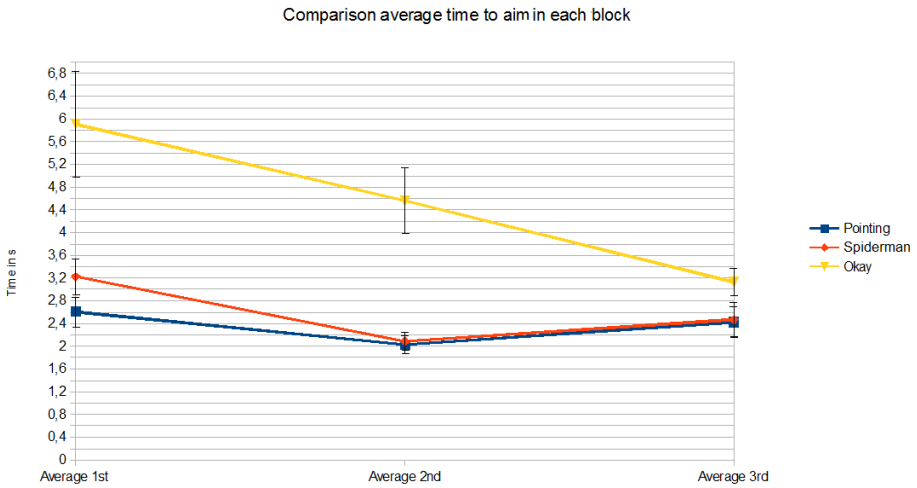


Figure 4.5: Average time to aim in each block sorted by the number of the trial

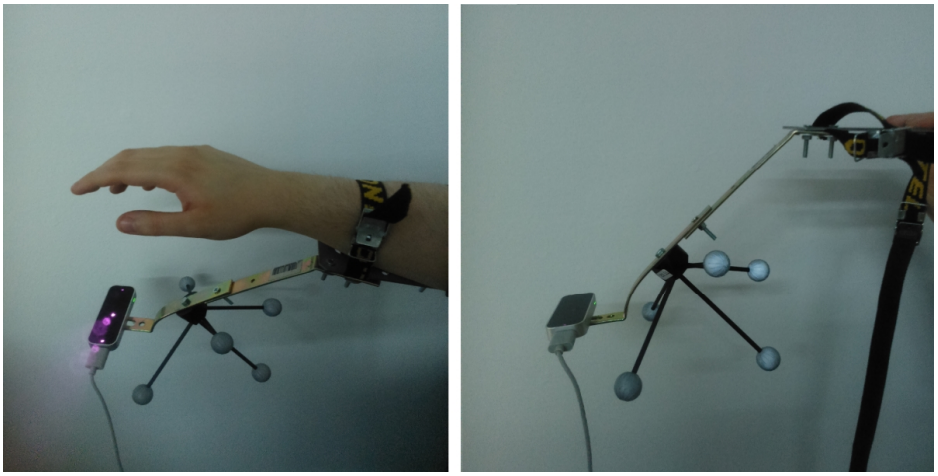


Figure 4.6: *The Leap Motion Controller* with metal construction, strapped to the hand (left), not strapped to hand (right)

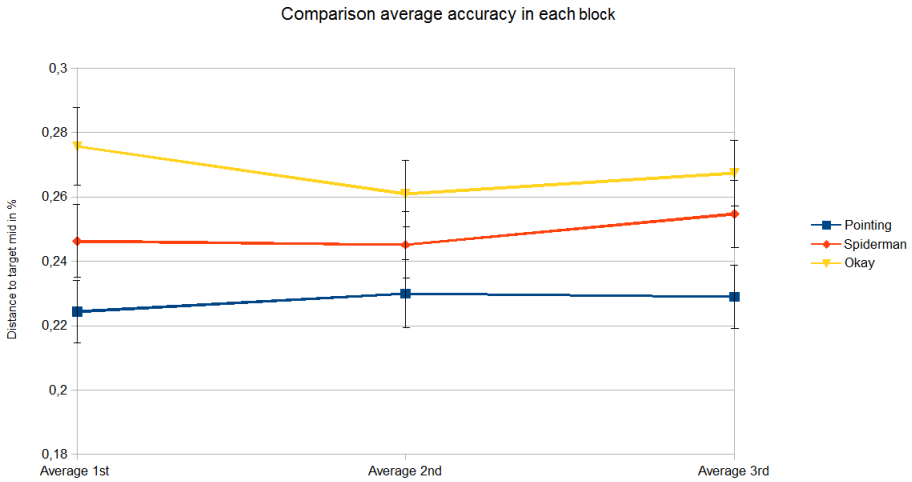


Figure 4.7: Average accuracy in each block sorted by the number of the trial

The higher targeting time also resulted in an equally higher overall time. The average time to target can be seen in Figure 4.8. The time to target with the *Pointing* posture and the *Spiderman* posture differed in about 0.2 seconds. In contrast to that the time to target with the *Okay* posture was, with an average of 4.5 seconds, almost double the time compared to *Pointing*. The time required to aim added up to an average total of 54.43 seconds per set, whereas the total time per set of the *Pointing* and the *Spiderman* posture only differed by 2.8 seconds on average (Figure: 4.9).

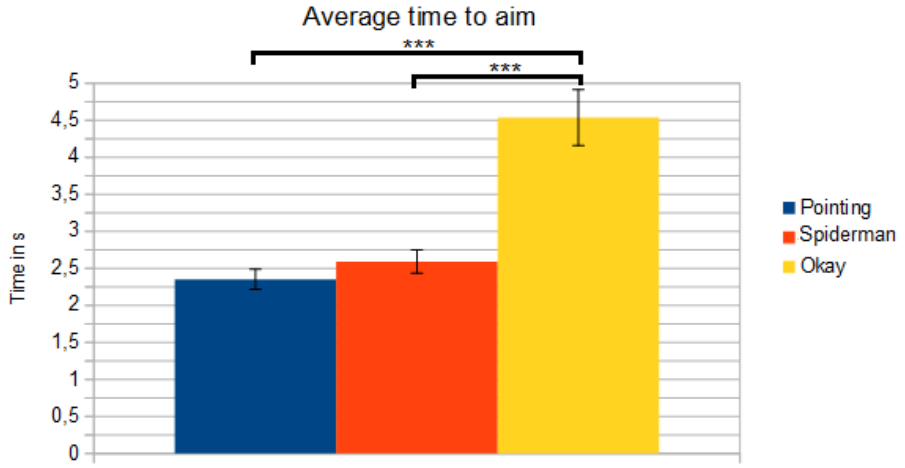


Figure 4.8: Average time to target

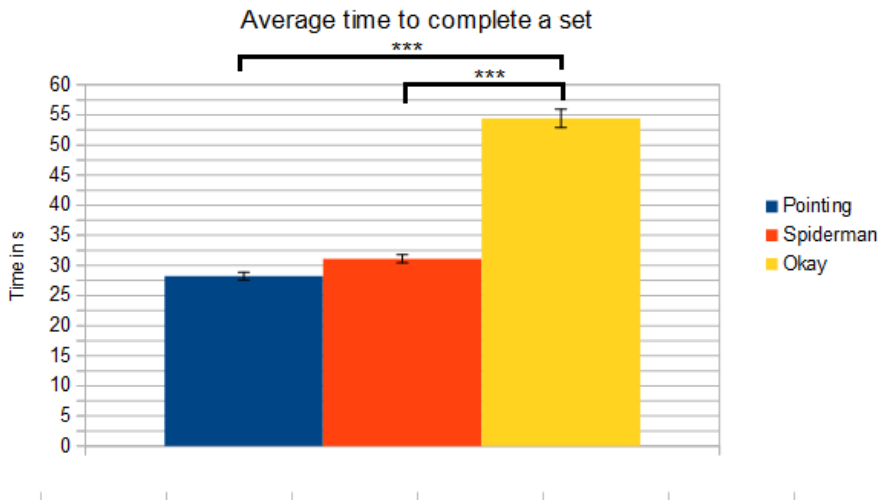


Figure 4.9: Average time to complete a set

In terms of deviance, both *Spiderman* and *Okay* postures did not perform better than the *Pointing* posture. The *Spiderman* achieved an average ratio of 0.248, which means that the users hit the inner quarter of the target on average. The *Okay* posture only achieved a deviance of 0.268 on average, which is outside the inner quarter of the target on average. The *Pointing* posture was even more accurate than the *Spiderman* posture, scoring an average 0.228 deviation (Figure: 4.10).

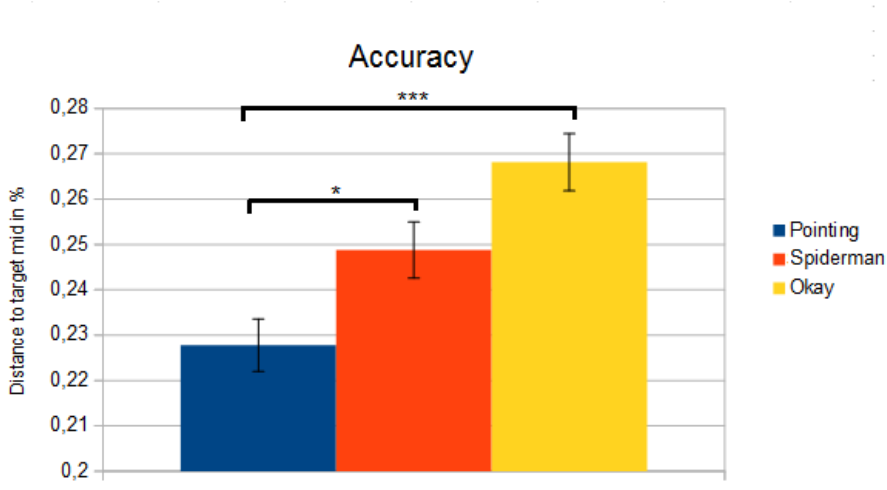


Figure 4.10: Average deviation

In general, the *Pointing* posture achieved the least deviance and the best "time to aim" values, since it was the most instinctive way for the subjects to point. The *Spiderman* posture seemed to be nearly as natural and comfortable as the *Pointing* posture due to the similarities in the time measurement and the low deviance. The *Okay* posture achieved the worst measurements of the tested postures and even after familiarizing with it, the subjects performed worse with it than with the other postures. Besides the overall performance of the different postures there were also some differences between the gender and the handedness of the subjects. As an example, the left-handed subjects performed better than the right-handed ones in time, but not in accuracy (Figure: 4.11).

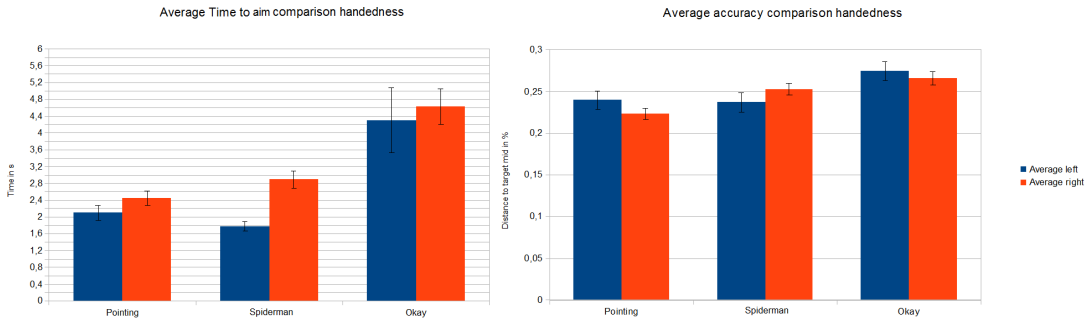


Figure 4.11: Performance comparing left handed and right handed subjects

Considering the gender, male subjects were faster when performing the sets. Especially the *Okay* and *Spiderman* posture had proven to be more difficult for female subjects, who needed almost twice as much time in average to complete a set in the *Okay* posture as male subjects while the deviance was also higher. For example, the distance to the target center was on average about 2 percent higher while pointing with the *Spiderman* posture (Figure: 4.12). The decreased accuracy and the higher aiming time could be explained by the weight of the metal contraption, since it might be more tiring for female subjects to wear than for male ones.

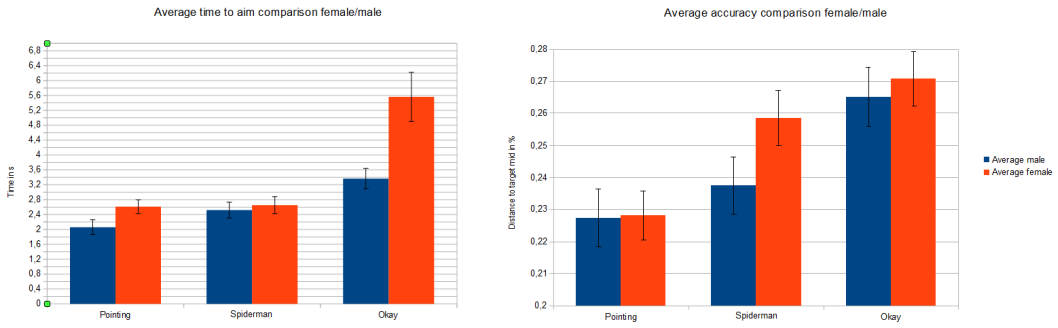


Figure 4.12: Performance comparing the gender of the subjects

4.6 Questionnaire

The main purpose of the questionnaire was to evaluate the subjects' opinion of the test and to find which posture felt the most comfortable and was the most accurate at the same time. About two thirds of the subjects rated the *Pointing* posture as the most accurate and comfortable posture, whereas about one third answered that the *Spiderman* was as pleasant as the *Pointing* posture. None of the subjects rated the *Okay* posture as the most accurate (Figure: 4.13). This posture was ranked the least comfortable and least accurate in total which also is supported by the results of the study. The subjects stated that the study benefited from the gamification aspect which did not only make the test more exciting, but even boosted the motivation of the subjects to continue despite hand fatigue caused by the weight of the metal construction attached to their forearm.

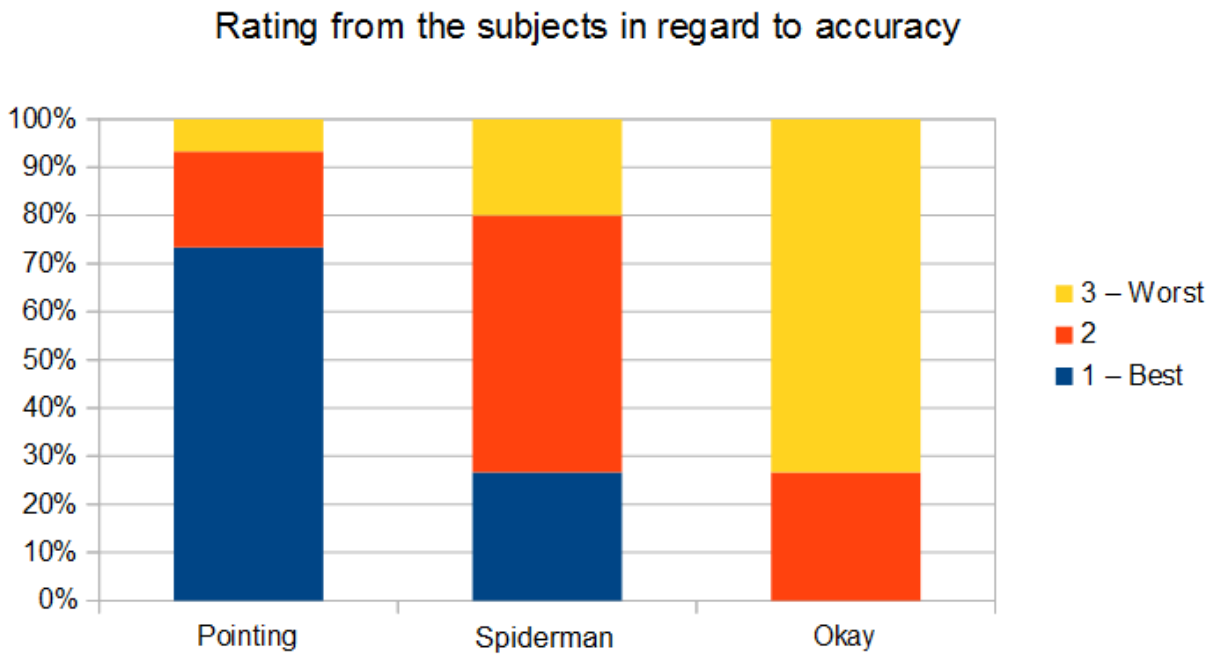


Figure 4.13: Results of the questionnaire in terms of accuracy, where rating 1 is best and 3 worst

5 Conclusion

The main goal of this thesis was to analyze several different postures which were suited for pointing in a three dimensional environment in terms of accuracy and time. To this end, a study was initiated. A total of three postures were selected, which were assumed to be the most promising candidates to complete pointing tasks. Those three were the *Pointing*, the *Spiderman*, and the *Okay* posture. The task in the study was to point at a total of twelve targets that were displayed in a random order. The data was collected during the study in order to analyze the user's posture performance according to overall time per set, the time it took to target, and the accuracy measured by the distance to the target center.
The evaluation of the data has shown that *Pointing* was the fastest and most accurate posture for pointing at a target, closely followed by the *Spiderman* posture which achieved almost as good results. The *Okay* posture was less suitable for pointing due to its unintuitive usage and the fact that it was not comfortable to stay in this posture for a longer period of time. Despite difficulties, a learning curve could be observed for the *Okay* posture. Due to hand fatigue, no learning effect for the other two postures could be found or confirmed. All in all, the findings indicate that some postures are more suitable for pointing in a three dimensional space than others. It turned out that the more instinctive and comfortable the posture is, the more accurate it is and the faster it can be applied. This opens a lot of opportunities in human to machine communication and a limitless usage of our hands as input device. Even though today's technology has its limits in terms of accuracy of hand tracking through hardware devices, a viable and reliable posture recognition is possible and already realizable.

6 Future Work

There is still a lot to improve in the field of posture recognition and research to be done to find a posture equally efficient as the standard pointing posture, which has more opportunities to take control of a diverse collection of technology and software. At first, the hardware would need to be improved since the *Oculus Rift* as well as the *Leap Motion Controllers* metal construction are too heavy to wear over a long period of time. Therefore, those would need to be made lighter and more comfortable to use. To reduce headache and dizziness of the subjects due to low resolution, framerate, and high latency, cameras attached to the *Oculus Rift* would also need to be improved. A better mobility of the subjects could be achieved by longer cable cords or even the use of wireless communication with the PC, which at the current state of technology can not be realized by any *VR HMD*.
Regarding the software, the dataset tracked for the kNN algorithm could be merged into a universal recognition file, since the more data is saved, the more users with differences in hand size can use the same data set without a recalibration of the database. In addition to that, the implementation of the hand postures could be improved by the insertion of the posture's position data into a dictionary instead of nested lists, providing a more flexible hand recognition system with more possible posture groups to work with.
The range of applications using different hand postures in *AR* are numerous, e.g. for entertainment in video games or in professional fields while using an user interface (*UI*) with posture control set in *VR*. Another approach could be to harness the possibilities of our hand postures and to implement a vast amount of different postures to improve the possibilities to use human to machine interaction or even a constant communication with a virtual agent.
One of future possibilities of this technology could be similar to the Wizard-of-Oz study with a drone where a subject gave orders to a drone via hand postures and gestures (Figure: 6.1). With mobile hand tracking devices, this could already be practicable [1].

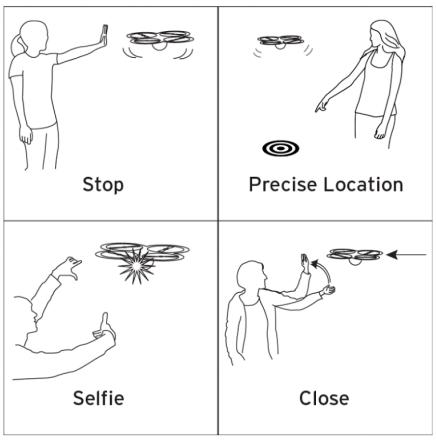


Figure 6.1: Human interaction during the Wizard-of-Oz study

7 Appendix A

Hardware

This section gives an overview of the hardware used in this study for hand tracking in a 3D environment.

The Leap Motion Controller

The *Leap Motion Controller* [2], or short *Leap*, is a hand tracking device which was released in 2013. It features three infra-red LEDs with a wavelength of 850 nanometers and two monochromatic infra-red (IR) cameras which are capable of observing a nearly hemispherical area. It is attached to the PC via an USB cable and is able to transmit about 300 images per second. These images consist of the reflected pattern-less IR-light generated by the LEDs. The *Leap* is being supported and constantly improved by *Leap Motion*. *Leap Motion* provides a development kit and API for game and software developers. In the project setup the *Leap Motion Controller* is fastened on a metal construction (see Figure: 4.6) which then is strapped to the forearm. Due to this positioning, the *Leap* is able to constantly track the users hand and prevents the user from leaving the tracking area. This fixation also provides the optimal distance between the fingers and the sensors of the *Leap*. An AR tracking target for the *ART* tracking system is attached to the metal construction in order to determine the fingertip position in the 3D AR room. The main task of the device is to track the hand and fingertip positions and send the analyzed data through the *Ubitrack framework* to the *Unity 3D Engine* (see Appendix B: 8.1).

Infrared cameras (ART System)

The experimental setup features four infra-red cameras, whose purpose it is to scan the project environment area and track augmented reality (AR) markers that are attached to several equipment in the tracked area. The position-data of the AR markers are then translated via the *Ubitrack framework* and passed to the *Unity 3D engine*, which uses the data to position finger objects in the virtual space according to the actual finger position tracked by the *Leap*. AR markers are attached to the *Leap Motion Controller*, the *Oculus Rift*, and a custom made portable calibration marker, whose functions will be described later in this paper. This camera setup was developed by *Advanced Realtime Tracking (ART)*[1].

[1] http://www.ar-tracking.com/products/, accessed 14.3.2016

Oculus Rift (DK2)

The *Oculus Rift*[2] is a head-mounted display which is used to project virtual targets, generated by the *Unity 3D engine*, into the users' field of view. Two cameras are installed on the front of the *Oculus Rift* whose purpose it is to record the surroundings and stream the video data onto the *Oculus Rift* display. To track the position of the *Oculus Rift* in the tracked area, the ART system combined with AR markers was used. The *Oculus Rift* setup is shown in Figure 7.1. The decision to use a head-mounted display (*HMD*) developed by *Oculus VR*, is based on the fact that they provide a SDK which supports the *Unity 3D Engine*.

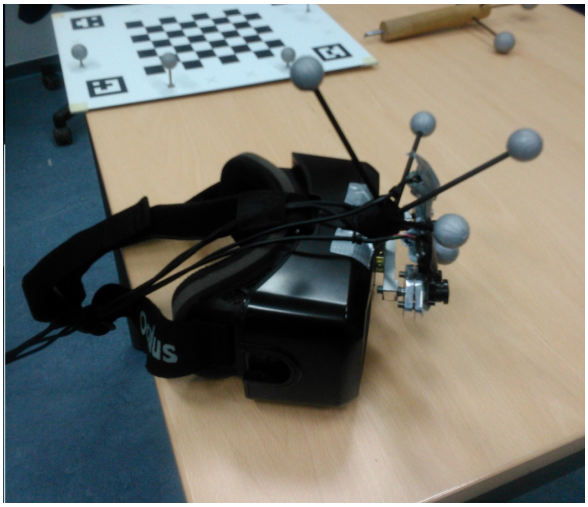


Figure 7.1: *Oculus Rift* setup with two front cameras and AR markers

[2]https://www.oculus.com/en-us/dk2/, accessed 14.3.2016

Portable calibration marker

The portable calibration marker (Figure: 7.2) is a custom made portable construction of AR markers in a uniquely identifiable configuration. This allows the accurate detection of its position and orientation in 3D-space, which can be used to calibrate the cameras recording the environment. Without this calibration it would be impossible to overlay the real and the virtual elements of the experiment. In addition, it was used to calibrate several devices in the tracked environment. In the project setup the position of the portable marker is tracked by the *ART system* and afterwards used by the *Ubitrack system* which converts the data to virtual positions in the *Unity 3D Engine*.

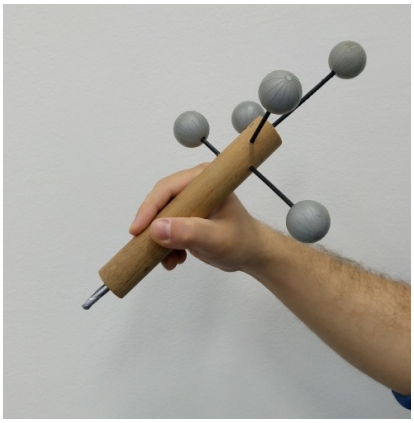


Figure 7.2: Portable calibration marker

The complete hardware setup is shown in Figure 7.3.

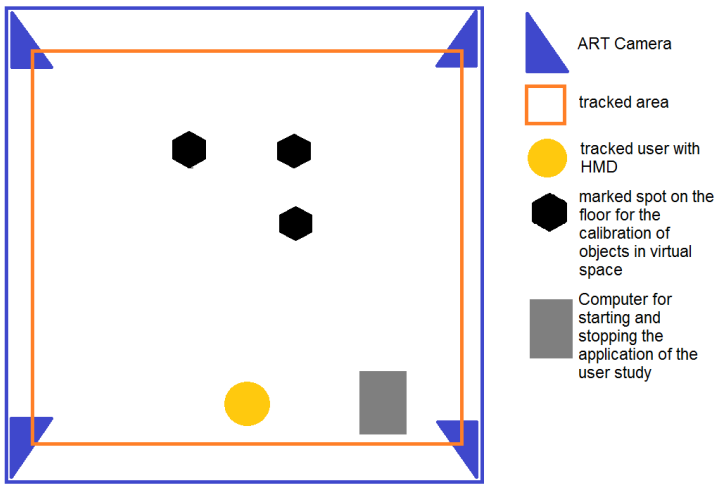


Figure 7.3: The complete hardware setup

8 Appendix B

8.1 Software and Algorithms

Unity 3D

One of the most commonly used and highly flexible game engines is the *Unity 3D Engine* which is capable of 2D and 3D rendering and can be used in both game and application development, using C# as main programming language. It is fully compatible with the *Oculus Rift* which is supported by *Oculus VR* with a SDK. It also provides a variety of assets that can be downloaded for free or bought from the *Unity Store*. This project makes use of the *Unity 3D Engine* as a development environment. It is the main software that is utilized to facilitate the communication between all hardware components and processes their output.

Ubitrack

This tracking framework is used to manage data. It was developed by the *Forschungsgruppe Augmented Reality (FAR)* at the Technische Universität München (TUM) and provides various drivers for several different types of tracking hardware as well as a vast amount of algorithms, calculating common calibration and tracking tasks[1]. It is also extensible for the use of different tracking devices like the *Leap Motion Controller*. In this project, it is managing the data of the *Leap*, *Unity 3D*, the *ART system*, and the *Oculus Rift*.

[1] http://campar.in.tum.de/UbiTrack/WebHome

8.2 Detailed Explanation of the Project Implementation

8.2.1 K-Nearest Neighbor

The k-nearest neighbor algorithm (kNN algorithm) is a basic classification algorithm in data mining that takes k elements or values into consideration and classifies data samples into different non-overlapping groups of values. Additionally, it can be used to train a system with the classification of new data. Storing all available cases and classifying new cases based on a similarity measure is the main task of the kNN. A small overview is given in Figure 8.1. To explain how the algorithm works in the project, and ease the explanation, the whole process is split into three phases.

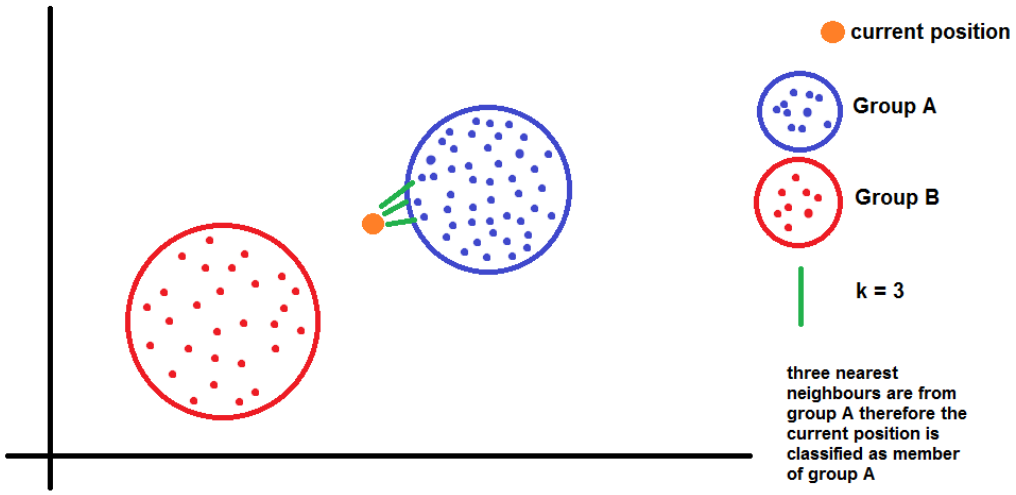


Figure 8.1: Simple visualization of the kNN algorithm in 2D

First Phase: Collecting Data

In the first phase a dataset needs to be created, which is split into several well described groups or cases. The higher the amount of data the more accurate the result of the kNN becomes, but the computation time and effort needed to classify a new sample increases accordingly. Therefore, the perfect balance of data to compare and accuracy that should be achieved needs to be found. The optimal number of nearest neighbors (k) is the square root of the overall number of samples.
In this project four hand postures for pointing have been defined. Each data group consisting of 150 values. The groups are the *Pointing*, *Spiderman*, *Okay* and Idle State posture (Figure: 3.2 and Figure: 3.1). The Idle State posture is needed to provide a neutral posture without a specific functionality for pointing to enable a fluent change of posture.

To gather all the values needed for the kNN algorithm to work, a script was implemented in *Unity 3D*, which is capable of storing the position data as three dimensional vectors in nested lists. The basic structure of a group is shown in Figure 8.2. Based on this structure each group contains 150 position values for each finger. Due to the maximization of the accuracy of the algorithm, each study participant had to calibrate and save their finger positions while taking the according posture for the specific group that should be calibrated. The calibration is lenient enough to make posture recognition possible across different subjects, albeit with a decrease in accuracy, due to subtle differences in the user's anatomy and hand poses.

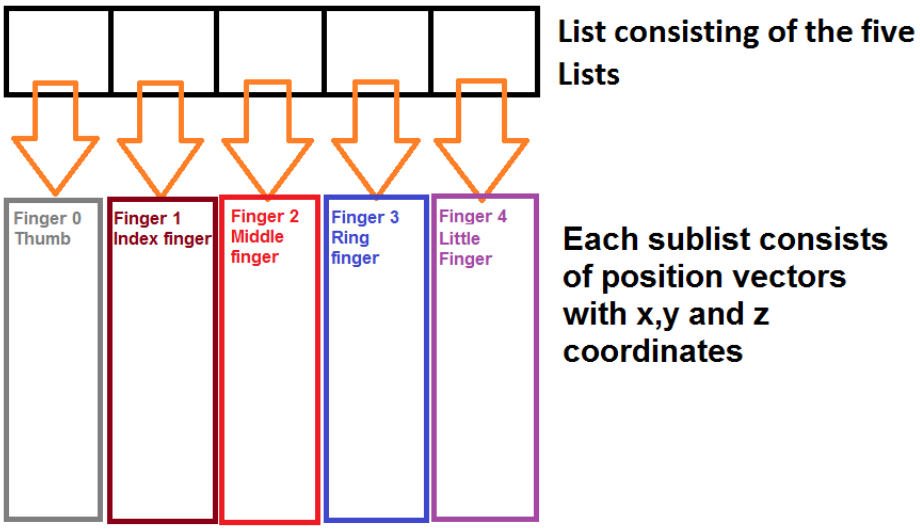


Figure 8.2: Structure of a group in the script

Second Phase: Calculations and Sorting

In the second phase of the kNN algorithm the closest distance to the current finger positions needs to be determined and saved in a new list, containing two dimensional vectors and which will be called shortest distance list in the following. The distance between the current finger positions and the saved finger positions in each group according to the finger index were calculated with the formula for euclidean distance (see Equation: 8.1) . The average distance values were saved in the x value while the y value was used to store an identifier according to the calculated group that then was added to the shortest distance list. For instance, when computing the distance of the pointing posture list and the current hand position, the euclidean distance between each element in the list and the current position is calculated. Then the average distance is saved in the x value and 0 as identifier[2] in the y value. The thereby created two dimensional vector is stored in a the shortest distance list. The sorting of this list after all posture distances values were determined, is essential for the evaluation of the kNN. The order provides a list beginning with the shortest distance and ending with the highest distance value to the current position.

$$\sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{8.1}$$

[2]The following identifiers were used: 0 for *Pointing*, 1 for *Idle State*, 2 for *Spiderman* and 3 for the *Okay* posture

Third Phase: Evaluation of the Distance Values

Using the list generated and sorted in the second phase, the kNN implementation evaluates the k first values in the shorted distance list according to the identifiers. K is defined as the number of nearest neighbors to the current position that needs to be evaluated.
Since the evaluation of the kNN algorithm in each frame requires a high amount of computational resources some optimizations were necessary.

8.2.2 Optimization of the kNN Algorithm

First of all, to decrease the time it takes to evaluate all data, adaptive classifications have been used. Similar to the basic incremental kNN algorithm several identifier counters were created [10]. Due to this, the result of the algorithm is either determined if k reaches the predefined ratio of nearest neighbors to a minimum of k samples, and therefore benefiting of an early break condition, or if the maximum of an identifier counter is reached. In this approach the shortest distance list is used instead of a R-tree queue. Another possibility to optimize the computational time and hardware resources the algorithm is either scheduling or parallelization. Two approaches were tried in this project, one of which was more efficient.

The first approach was to use a coroutine which then evaluates the nearest neighbors according to an frame independent scheduling and therefore splitting the calculation of the kNN algorithm over several frames. The assumption was, that since the posture was static in the study, the algorithm would be evaluated some frames after assuming the posture. The implementation of coroutine is able to run smoothly between 250 and 1000 frames per second (FPS), with some peaks at about 120 FPS when the posture was changed. Figure 8.3 shows the FPS of the coroutine based kNN algorithm with three different postures recognized and without being connected to the AR system. Even though the split of the calculation into several frames the coroutine was not able to keep up with the data input when extending the setup with the *Oculus Rift* and another posture, hence causing the application to crash. Therefore a more powerful method was needed to be implemented.

The second approach was the parallelization of the kNN algorithm using threads. Through this the time for the evaluation of the kNN algorithm was more than doubled. It was also possible to handle more postures and the data input in the AR environment. To improve the performance even more, the kNN algorithm will not be evaluated with data from the next frame if the data of the current frame was not already interpreted. The FPS still dropped when changing a posture, but it's severity was strongly decreased, so the response time for a successful posture recognition was decreased to less than one millisecond. As seen in Figure 8.4 the performance of the project without the AR environment is pending between 3000-4000 FPS and works with an almost constant 1000 FPS with the AR environment, needing less than 0.4 ms for an evaluated posture recognition.

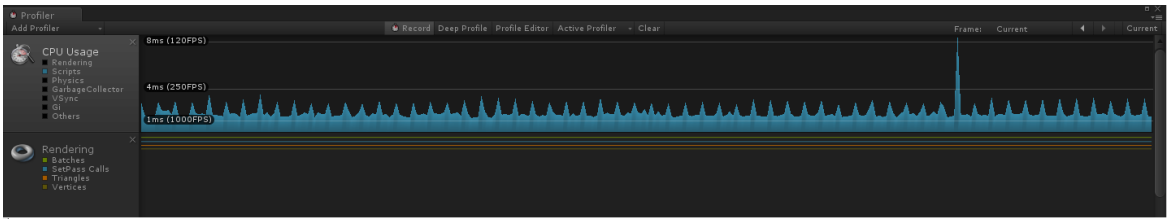


Figure 8.3: Performance of the project setup running with coroutines, using three different postures, without AR environment

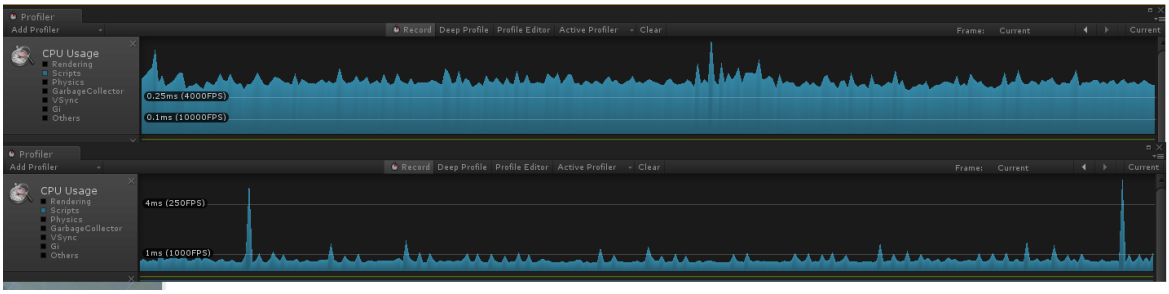


Figure 8.4: Performance of the project setup running with threaded kNN algorithm, using four different postures, without AR environment (top), performance of the project setup running with threaded kNN algorithm, using four different postures with AR environment running (bottom)

List of Figures

Bibliography

[1] Jessica R Cauchard, LE Jane, Kevin Y Zhai, and James A Landay. Drone & me: an exploration into natural human-drone interaction. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 361–365. ACM, 2015.

[2] Alex Colgan. How does the leap motion controller work?, 2015.

[3] Dejan Chandra Gope. Hand tracking and hand gesture recognition for human computer interaction. *American Academic & Scholarly Research Journal*, 4(6):36, 2012.

[4] Nicholas Katzakis, Masahiro Hori, Kiyoshi Kiyokawa, and Haruo Takemura. Plane-Casting: 3D Cursor Control With A Smartphone. In *The 3rd Dimension of CHI (3DCHI): Touching and Designing 3D User Interfaces workshop at ACM CHI*, pages 13–21. ACM, 2012.

[5] Nicholas Katzakis, Kazuteru Seki, Kiyoshi Kiyokawa, and Haruo Takemura. Mesh-Grab and Arcball-3D: Ray-based 6-DOF Object Manipulation. In *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, pages 129–136. ACM SIGCHI, 2013.

[6] Nicholas Katzakis, Robert Jl Teather, Kiyoshi Kiyokawa, and Haruo Takemura. INSPECT: Extending Plane-Casting for 6-DOF Control. *Human-centric Computing and Information Sciences*, 5(22), 2015.

[7] Pedro Lopes, Alexandra Ion, Willi Mueller, Daniel Hoffmann, Patrik Jonell, and Patrick Baudisch. Proprioceptive interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 939–948. ACM, 2015.

[8] Mathieu Nancel, Julie Wagner, Emmanuel Pietriga, Olivier Chapuis, and Wendy Mackay. Mid-air pan-and-zoom on wall-sized displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 177–186. ACM, 2011.

[9] Takayuki Ohnishi, Nicholas Katzakis, Kiyoshi Kiyokawa, and Haruo Takemura. Virtual interaction surface: Decoupling of interaction and view dimensions for flexible indirect 3D interaction. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 113–116. IEEE, 2012.

[10] Stefanos Ougiaroglou, Alexandros Nanopoulos, Apostolos N Papadopoulos, Yannis Manolopoulos, and Tatjana Welzer-Druzovec. Adaptive k-nearest neighbor classification based on a dynamic number of nearest neighbors. In *ADBIS*, volume 7, pages 66–82. Citeseer.

[11] Yves Rossetti, Cedric Meckler, and Claude Prablanc. Is there an optimal arm posture? deterioration of finger localization precision and comfort sensation in extreme arm-joint postures. *Experimental Brain Research*, 99(1):131–136, 1994.