TU WIEN Informatics

A Virtual Reality Interface for Teleoperating Mobile Robots in Exploratory Tasks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Martin Crepaz, BSc
Matrikelnummer 11776187

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Mitwirkung: Projektass. Dr. Francesco De Pace

Wien, 28. April 2025

Martin Crepaz	Hannes Kaufmann

TU WIEN Informatics

A Virtual Reality Interface for Teleoperating Mobile Robots in Exploratory Tasks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Martin Crepaz, BSc
Registration Number 11776187

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Assistance: Projektass. Dr. Francesco De Pace

Vienna, April 28, 2025

Martin Crepaz	Hannes Kaufmann

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Martin Crepaz, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 28. April 2025

Martin Crepaz

Danksagung

Besonderer Dank gilt meinen Betreuern Hannes Kaufmann und Francesco De Pace, welche es mir ermöglicht haben, an diesem faszinierenden Thema zu arbeiten. Für die kontinuierliche Unterstützung und wertvollen Ratschläge, während der gesamten Zeit bin ich sehr dankbar. Ebenso gilt mein Dank Hugo Brument, der mir bei der Vorbereitung der Nutzerstudie beratend zur Seite gestanden ist.

Mein aufrichtiger Dank gilt auch meinen Eltern, die mir durch ihre Unterstützung den Freiraum gegeben haben, mich voll auf meine Ziele zu konzentrieren. Meiner Schwester danke ich besonders für ihre wertvollen Tipps und ihr aufmerksames Auge beim Korrekturlesen dieser Arbeit.

Außerdem möchte ich mich bei allen TeilnehmerInnen der Nutzerstudie bedanken, die durch ihre Teilnahme und konstruktiven Rückmeldungen wesentlich zum Erfolg der Arbeit beigetragen haben.

Schlussendlich möchte ich ein Dankeschön an alle aussprechen, die mich während meines Studiums begleitet und unterstützt haben.

Acknowledgements

Special thanks go to my supervisors Hannes Kaufmann and Francesco De Pace, who made it possible for me to work on this fascinating topic. Their continuous support and valuable advice during the entire time made the realization possible. I would also like to thank Hugo Brument, who advised me on the preparation of the user study.

My sincere thanks also go to my parents, whose support gave me the freedom to concentrate fully on my goals. I would especially like to thank my sister for her valuable advice and her attentive eye when proofreading my work.

I would also like to thank all the participants in the user study, whose participation and constructive feedback contributed significantly to the success of this work.

Finally, I would like to thank everyone who has accompanied and supported me during my studies.

Kurzfassung

Human-Robot Interaction (HRI) beschäftigt sich mit der Zusammenarbeit von Mensch und Roboter, welche durch die fortschreitende Automatisierung immer mehr an Bedeutung gewinnt. Die stetige Entwicklung in der Robotik eröffnet immer mehr Einsatzmöglichkeiten, wohingegen die Steuerung durch den Menschen weiterhin eine Hürde darstellt. Insbesondere die Fernsteuerung von Robotern durch den Mensch über Distanzen hinweg, erfordert besondere Mittel.

Die Darstellung der Roboter-Umgebung in einer anschaulichen Form stellt dabei eine zentrale Herausforderung dar. Traditionelle Methoden über 2D-Bildschirme für die Anzeige der Inhalte, sowie die Steuerung über Maus und Tastatur stoßen dabei an ihre Grenzen. Um diese Herausforderungen zu überwinden, benötigt es immersive Technologien wie Virtual Reality (VR), Augmented Reality (AR) und Mixed Reality (MR), welche innovative Ansätze für die Visualisierung und Steuerung ermöglichen.

In dieser Arbeit wird ein System zur Fernsteuerung eines mobilen Roboters in einer realen Umgebung mittels VR vorgestellt. Grundlage hierfür ist eine Echtzeit 3D-Rekonstruktion aus den Bilddaten der am Roboter montierten Kamera. Die Darstellung der Umgebung in VR wurde mit der Unity3D Spiel-Engine entwickelt. Es wurden zwei Navigationsmetaphern implementiert, die den ausführenden Personen ermöglichen, einen physischen Roboter in der virtuellen Darstellung der realen Umgebung zu steuern.

Eine Benutzerstudie wurde durchgeführt, um die Performance, die Benutzerfreundlichkeit und die Intuitivität der beiden Metaphern zu erheben und zu vergleichen. Die Aufgabe der Benutzerstudie bestand darin, den Roboter zu navigieren und die Umgebung nach dem Ziel abzusuchen.

Die Ergebnisse zeigten signifikante Unterschiede in der objektiven Performance, aber nahezu keine Abweichung in der subjektiven Wahrnehmung. Beide Metaphern erwiesen sich als geeignet für die Fernsteuerung eines mobilen Roboters in VR.

Abstract

Human-robot interaction (HRI) deals with the interaction between humans and robots, which is becoming increasingly important as automation progresses. The ongoing development in robotics is opening up more and more possible applications, whereas human control continues to pose a challenge. In particular, the remote control of robots by humans over distances requires special means.

The visualization of the robot's environment represents a central challenge. Traditional methods using 2D screens to display content and mouse and keyboard to control the robot reach their limits. Overcoming this challenge requires immersive technologies such as Virtual Reality (VR), Augmented Reality (AR) or Mixed Reality (MR), which enable innovative approaches to visualization and control.

This thesis presents a system for teleoperating a mobile robot in a real environment using VR. The basis for this is a real-time 3D reconstruction from the image data of a camera mounted on the robot. The system for visualizing the environment and controlling the robot in VR was developed using the Unity3D game engine. Two navigation metaphors were implemented to enable human operators to control a physical robot in the virtual representation of the real environment.

A user study was conducted to assess and compare the performance, usability, and intuitiveness of both metaphors. The task of the user study was to navigate the robot and to investigate the environment to find the target point.

The results showed significant differences in objective performance, but almost no deviation in subjective perception. Both metaphors proved to be suitable for the teleoperation of a mobile robot in VR.

Contents

CHAPTER 1

Introduction

The interaction between humans and robots is a central core area of scientific research. Its importance continues to increase due to further automation. The field of study that deals with the understanding between humans and robots is called Human-Robot Interaction (HRI) and is defined by Goodrich and Schultz [1] as follows:

> *Human–Robot Interaction (HRI) is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans.*

When humans and robots interact with each other, communication is required, which can take various forms and differ depending on the proximity between humans and robots. A distinction is made between the categories of remote interaction and proximate interaction [1]:

- Remote interaction refers to the temporal and spatial separation from each other. In the context of mobile robots, remote interaction is commonly described as teleoperation or supervisory control. In this work the term teleoperation is used.
- Proximate interaction, on the other hand, means that the robot and the human are co-located.

Since the goal of this thesis is to control a mobile robot from a distance, the focus of this work lies on the concept of teleoperation. Research in this area has shown that robot teleoperation is associated with significant challenges and therefore requires a high level of user training and skills [2]. The maintenance and intervention of remote control systems are particularly important in hazardous environments where conditions are high risk. Advanced environmental awareness and collision avoidance are particularly important for safe and efficient teleoperation [3].

Research on robot remote control is of particular importance as it allows robots to be used in dynamic and complex environments.

Examples for mobile robot teleoperation applications are:

- rescue operations [4, 5]
- dismantling drug labs [6]
- nuclear monitoring [7]
- bomb disposal [8]
- mine discovery [9]

Robot teleoperation is essential for effective navigation, especially in dynamic and hazardous environments. In certain situations, robots are the only possibility for entering a facility, for example, when there is a high danger of nuclear radiation for humans [7]. In contrast to autonomously operated robots, human-operated robots benefit from the combination of human flexibility, cognitive performance, and soft skills with the capabilities of robots. Therefore, it is important to find ways for a smooth and intuitive communication between humans and robots [10].

Traditional approaches for robot teleoperation rely on 2D displays for visual feedback, as well as common input methods such as mouse/keyboard, gamepad [11, 12], joystick [13], gestures [2, 14] and voice commands [15, 16]. Human operators should be able to fully concentrate on the environment where they have to navigate the robot, therefore, an intuitive control system is crucial.

Conventional interfaces that are based on 2D methods for visualization make it difficult for operators to interact with the 3D environment [17]. This is because the visualization on 2D screens leads to a reduced perception of depth. This makes it difficult for operators to react to changes in real-time and to use the natural ability of humans to manipulate objects in 3D [18].

Traditional interfaces also restrict situational awareness (SA). Situational awareness describes the understanding of the robot environment, the position of the robot, the contextual movement of the robot in the environment, and the prediction of its future behavior [9, 18, 19].

As Endsley [20] states in his work about situational awareness in pilot/system performance of aircraft: "*Even the best trained and most experienced pilots can make the wrong decisions if they have incomplete or inaccurate SA.*". This example emphasizes the importance of situational awareness and can also be applied to the human/robot interaction.

Previous research has also suggested that traditional keyboard and monitor interfaces lead to a higher cognitive load and lower usability compared to more innovative interfaces [21]. These limitations can be overcome by using immersive VR interfaces, which are

characterized by freedom of view and a natural method of control. VR interfaces shorten the time required to complete tasks and thus increase the operator's performance [19, 22]. VR has proven to be a promising technology, as immersion is a central concept of VR. VR allows the user to view the virtual environment as if the user was actually in that world [2].

Immersive interfaces are often based on the use of head-mounted displays (HMDs), which make it possible to combine virtual and real content seamlessly and create an immersive user experience. Head mounted display systems provide tracking for headset and gaming controllers, which allows the user to easily interact with the virtual environment and its objects. Rendering the robot's environment on an immersive display has been demonstrated to enhance the operator's situational awareness and performance of spatial tasks [18]. These benefits arise from an extended field of view and the user's ability to control the orientation of the camera trough head movements, which is more intuitive than joystick or other control methods [23]. A study comparing VR-based and video-based robot teleoperation found that 3D visualization in VR ensures a better self-localization, terrain assessment, view controlling, navigation around corners and obstacles avoidance [11].

A study comparing VR and traditional teleoperation interface designs, where trained robot operators work in simulated nuclear monitoring, suggests that VR operators experience lower objective cognitive workload compared to operators using traditional interfaces [7].

Challenges of using HMDs in interface design compared to the use of computer monitors are the effects of simulator sickness such as headaches or nausea [18]. Moss and Muth [24] mention that these effects can be reduced by increasing HMD frame rate or providing the use with postural support like railings to hold on for a secure standing.

Summing up the above mentioned benefits of immersive interface outweigh possible limitations. Because of that and since it was decided to build on a fully immersive experience in this thesis, VR was selected as the interface method.

1.1 Motivation & Problem Statement

The use of immersive technologies to remotely control robots has particular advantages when it comes to perceiving the environment. Studies have shown that this leads to a significantly higher level of situational awareness. Previous studies on robot control have focused mainly on comparing traditional approaches with systems developed with immersive technologies. The traditional approaches, which serve as a basis, use monitors for visual representation and mouse and keyboard for control [2, 11].

There are several approaches for visualizing the environment for the user. One approach would be a live camera feed, which streams the ego-centric perspective of the robot camera to the user. Another possibility is the 3D visualization with point clouds, representing the scene via rough reference points to get a feeling of the environment. This approach provides a broad overview but lacks details for the intuitive perception of a space [2, 25].

This work aims to close the gap of limited 3D visualizations and to achieve an advanced visualization using a real-time 3D mesh of the environment. This approach goes beyond the purely visual representation through camera streams and point clouds and offers the user a detailed representation of the environment. This makes it easier for the user to orientate themselves in the virtual environment and navigate the robot precisely [18].

1.2 Aim of the Work

The aim of this thesis is the development of a software system to enable human operators to remotely explore an environment by controlling a real robot in VR. This work includes the exploration of the visualization possibilities of a real environment in a virtual form. This enables the user to get as much information as possible and immersing into the virtual world. By enabling the user to control the movement of the robot and thus navigate it through the scene, the possibilities of navigation are also investigated.

A user study will be conducted to compare two different navigation metaphors and evaluate the usability, intuitiveness and efficiency of the different approaches. Finally, a recommendation is given for the most suitable interaction option for controlling a robot in VR.

The basis for the system is a Boston Dynamics Spot robot, which is equipped with a camera that can record color and depth images. These images are processed through a 3D reconstruction pipeline to create a real-time 3D mesh of the robot's environment. This mesh is visualized for the user in the VR environment, allowing the user to observe the robot's surroundings and navigate the robot in the scene.

The results of this work are intended to contribute to research on innovative navigation metaphors in the field of Human-Robot Interaction.

1.3 Structure of the Thesis

The thesis is structured as follows: Chapter 2 provides an overview of the theoretical background of the work and the knowledge that has been provided in the past on this area of research. Chapter 3 deals with the design and implementation of the system and provides an overview of the process from image capture to 3D reconstruction and visualization to the user. In addition, implementation of the control methods is discussed and presented in more detail. Chapter 4 presents the process of the user study and shows the results. Chapter 5 discusses the performance and limitations of this thesis. Finally, Chapter 6 summarizes the work and discusses possible future work.

CHAPTER 2

Related Work

This chapter presents a theoretical background of the current state of art and the concepts that are relevant to this work on human-robot interaction.

The teleoperation of a robot is a challenging task. It can, for example, be difficult for the user to know which commands are required to reach a target. Furthermore, the user may experience difficulties anticipating how a control input will affect the system. In addition, the user has to navigate the robot to evaluate its surroundings, which could possibly lead to unpredictable or hazardous situations for the robot itself or humans nearby. These problems occur because, with teleoperation, the mapping between the user's command and the robot dynamics is hidden from the user. The user needs to learn through experience how precisely an action affects the robot's movements. This means that the operator has to learn indirectly from practicing with the input controller and continually verify the effects on the robot [26].

There are multiple options for teleoperating a robot using different input modalities. The following sections emphasize introducing traditional interfaces and immersive technologies with a focus on the theoretical background that has been provided by other researchers.

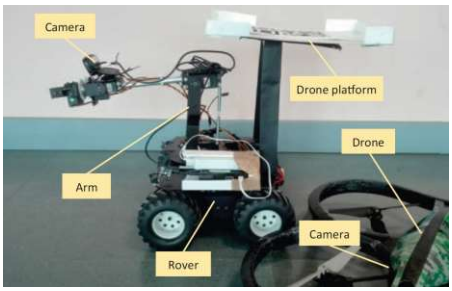
2.1 Traditional Teleoperation Methods

Numerous options are available for the teleoperation of robots, ranging from physical input devices such as joysticks, gamepads, or keyboards to technologies such as hand tracking or voice control. This variety enables flexible and application-oriented control, adapted to individual requirements [3, 27].

2.1.1 Physical Input Devices

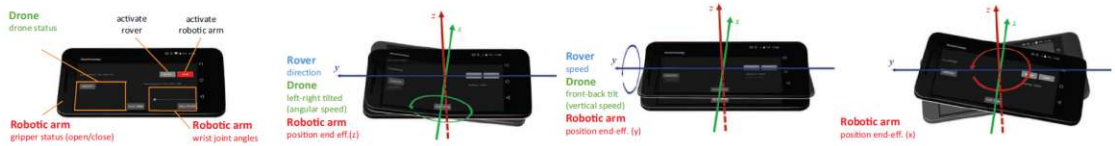
Bonaiuto et al. [28] investigate how different user interfaces are suitable for controlling multiple robots (see Figure 2.1b). The interfaces tested are a gamepad (see Figure 2.1a),

a mobile device (see Figure 2.1c), and a hand tracking system. The aim of their work is to identify which interaction method is best suited for control.
In a user study, the three control methods are compared in terms of their performance and usability. The participants had to control several robots to solve a search and selection task. They combined the abilities of a drone, a rover, and a robotic arm. The task includes several steps such as controlling the rover, flying the drone to find an object, and finally using the robotic arm to lift an object. The results show that the tasks were solved fastest with the mobile device. It is also clear that certain input methods are particularly suitable for certain robots and subtasks. The hand tracking system was particularly convincing when it came to controlling the robot arm. In terms of user-friendliness, the gamepad was selected by the participants as the preferred interface.



(a) Robot system with rover, robotic arm and drone

(b) Gamepad interface



(c) Mobile Device interface

Figure 2.1: Robot system and control interfaces by Bonaiuto et al. [28].

2.1.2 Gesture

The recognition of hand gestures can serve as an alternative to the physical input devices currently used in Human-Computer Interaction (HCI). It can simplify the learning of sophisticated control systems as it offers an intuitive system for humans to interact with a computer. Gesture control uses body movements, usually in the form of hand signals, to communicate messages to the system [29].

Paterson and Aldabbagh [30] divided the interpretation of human hand gestures into two main types: The Data Glove method and the Computer Vision method to recognize hand gestures. In the Data Glove method, the user wears a special glove. The glove is equipped with acceleration sensors and gyroscopes, as well as a power source for wireless versions or a network of data and power lines for wired ones. However, data gloves are usually expensive to buy and uncomfortable for long-term use, and are also not robust

enough for use outdoors. In the other approach, based on computer vision, the hand is observed in isolation in order to track its movements. The advantages of this method include the use of cameras, which makes the method easier to use because of the better availability of cameras nowadays. Today, every mobile phone is equipped with one or more cameras. In addition, cameras do not interfere with hand movement, and the use of a computer vision system makes it possible to monitor multiple hands.
In experiments, the Data Glove method showed higher accuracy for gesture recognition than some methods using computer vision. However, the disadvantages of higher overall costs and poorer comfort must also be taken into account when designing a system with hand gesture recognition [30].

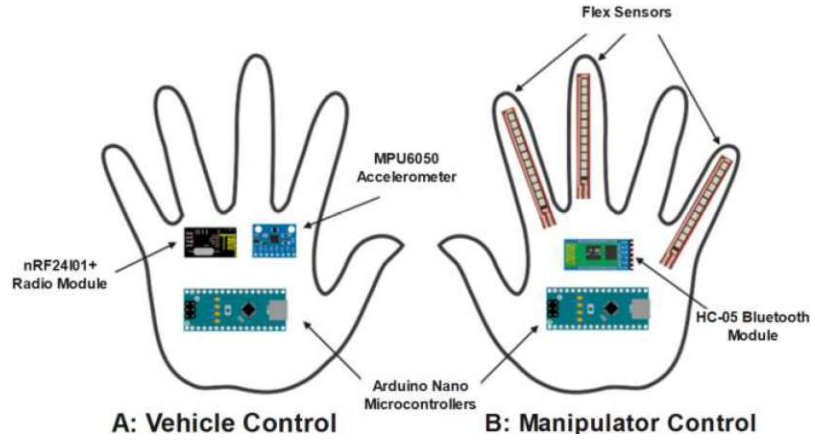
Solly and Aldabbagh [14] introduce a 3D printed maneuverable robot that is remotely controlled by gestures. Two glove controllers (see Figure 2.2a) were used simultaneously to control the robot vehicle and a robotic manipulator. With the hand-worn glove controllers, the 5-axis robotic manipulator and a robot vehicle can be controlled by hand gestures. The left hand controls the robot vehicle (see Figure 2.2b), while the right hand controls the robot manipulator, which is mounted on the robotic vehicle. The experiment consisted of a pick-and-place task, which demands precise navigation to pick up an object and deposit it at a designated position. The study shows that gesture-controlled robotics offers promising opportunities to improve human-robot interaction. This provides a more natural and intuitive way for the operator to communicate with the robot. The controllers enable more precise control, as even small movements of the hand are recognized and converted into robot movements.

In industrial applications, gesture control can be used to move objects from one position to another. The control method attempts to mimic human arm movements as closely as possible, allowing the human to control the robot from a safe environment [14].

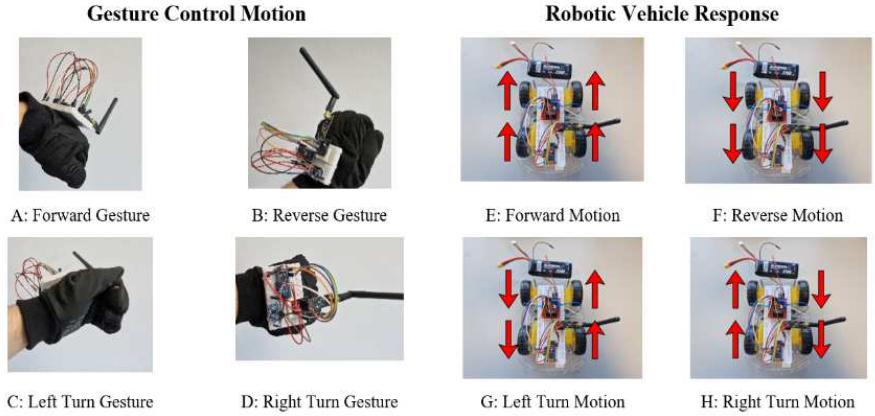
Other studies combine hand tracking with immersive technologies such as mixed reality and virtual reality. This combination improves the user-friendliness, efficiency, and effectiveness of teleoperation compared to using traditional control methods such as physical controllers only [2].

2.1.3 Voice

The most popular form of human communication is the voice. Therefore, speech recognition is a preferred choice for service robots, for example. With a microphone, sounds and voice signals can be transformed into electrical signals. Speech signals are translated into text form by speech recognition to provide instructions for computers. Robots that are controlled by speech understand thousands of commands and execute them. Due to the unique speech patterns of each person, voice recognition represents a complex challenge. As a result of continuous development in the field of artificial intelligence, considerable improvements have been achieved. The use of robots with voice control ranges from manufacturing to use in hospitals for delivering medication or monitoring corridors [15].



(a) Design of gloves



(b) Gesture Control Mapping

Figure 2.2: Gesture Control for controlling robotic manipulator and robot vehicle [14].

In their work, Ahmad et al. [15] focus on the control of a mobile robot platform using voice commands. For an overview of the system design, see Figure 2.3. The voice commands are converted into movement commands using speech recognition software, which is then used to control a mobile robot. The system consists of a microcontroller for controlling the motors of the mobile platform, among other things. The voice commands are recorded via a wireless headset microphone and transmitted to the virtual control assistant. Speech recognition software processes the voice commands to perform the corresponding action. The speech recognition software is trained in advance so that the customized voice commands are recognized. The performance of seven voice commands was evaluated in an experiment. The evaluation of the proposed prototype showed that the voice-controlled mobile robot platform that was developed has proven efficient in executing commands. This enables a robot to be controlled by voice instead of a joystick controller or keyboard.

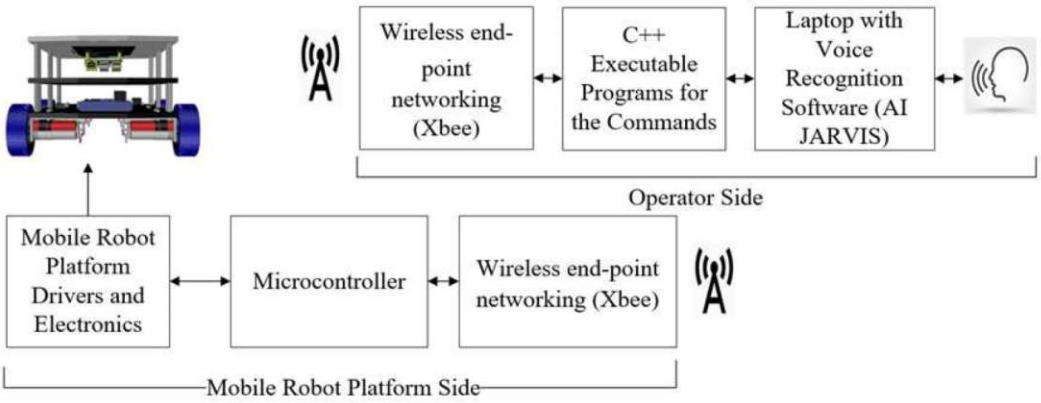


Figure 2.3: Overview of the system design for controlling a mobile robot platform using voice commands by Ahmad et al. [15].

In their paper, Poncela and Gallardo-Estrella [16] present the development of a user-dependent acoustic model for the Spanish language, which provides teleoperation of a robot platform via the user's voice. The development focuses on creating a new speech recognition system for Spanish speakers, based on a customized acoustic model for remotely controlling a robot using a series of commands. The results show a high recognition rate of speech commands and a successful navigation of the robot. Another benefit of the system was that it could be easily adapted to new grammars and platforms, which makes its a solid basis for further developments in the field of teleoperating a robot with voice commands.

This section gave an overview of some traditional interfaces for teleoperating a mobile robot. A comparison of controlling a hyper-redundant robot by traditional and immersive interfaces concluded that the latter provides improved visual feedback, efficiency, and situational awareness [31]. With these findings in mind, the following section elaborates on the topic of mixed reality interfaces.

2.2 Immersive Teleoperation Interfaces

Milgram and Kishino [32] introduce the concept of the reality-virtuality continuum (see Figure 2.4), the spectrum of which ranges from the complete real environment to a completely virtual environment. Augmented reality refers to the extension of the real world with virtual elements. Mixed Reality (MR) describes the range between reality and virtuality, in which real and virtual objects are combined. MR includes both AR and Augmented Virtuality (AV). AV augments the virtual environment with real-world objects. VR at the end of the continuum comprises a completely computer-generated virtual world and is not part of MR in the definition.

In VR, the environment consists only of virtual objects [33]. Aligning the real world and virtual objects has been a challenging task in the field of MR and has led to slower

progress of its development compared to VR in the past years. Lately, however, VR/AR headsets have improved significantly, which could be beneficial in the future evolution of MR [9].

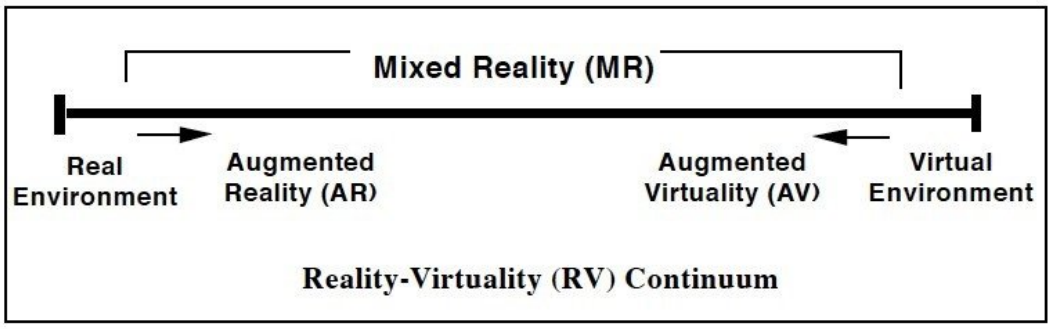
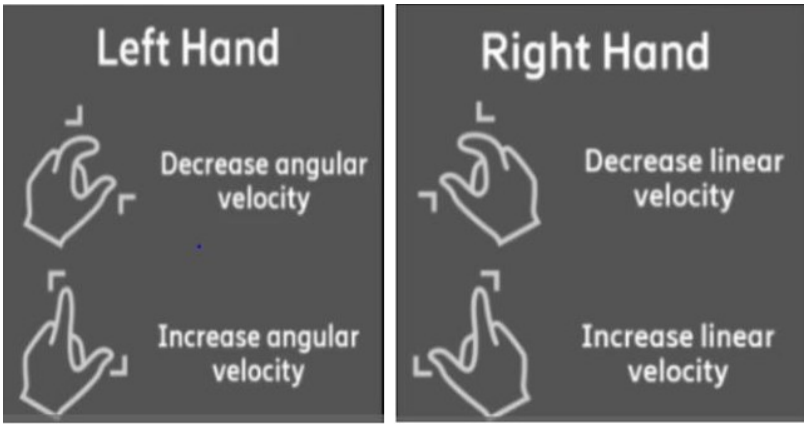


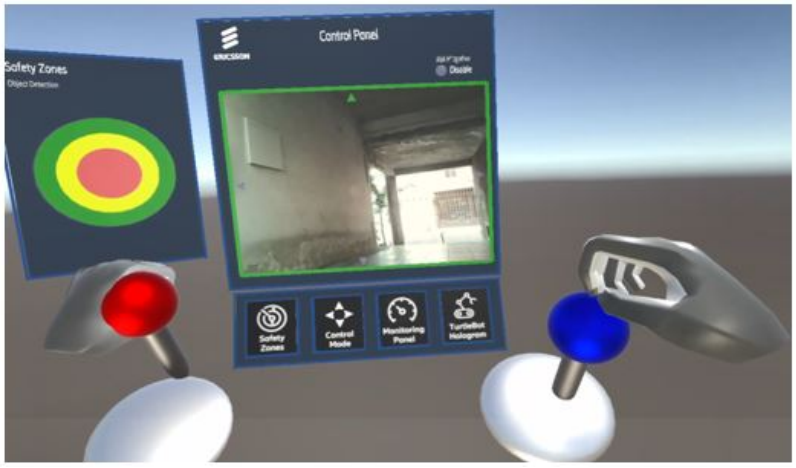
Figure 2.4: Reality-Virtuality Continuum, illustrating the spectrum between real environment and virtual environment proposed by Milgram and Kishino [32]. Image taken from [34].

In their paper, Batistute et al. [2] compared the robot teleoperation efficiency between three different methods: a traditional control method, and two immersive control methods. A physical controller in the form of a keyboard is used as a traditional interface. A mixed reality control system and a virtual control system are used as immersive control systems. The mixed reality setup uses a mixed reality headset (Magic Leap One,) which has hand movement tracking. The left and right hands (see Figure 2.5a) control the robot's speed and the angular velocity. The respective speed is changed with the finger distance. The virtual reality setup consists of a virtual reality headset, which is equipped with a sensor device for hand tracking. The sensor device tracks the operator's hand movements.
The robot is controlled via virtual joysticks (see Figure 2.5b) located in the virtual environment. While the left joystick controls the robot's speed, the right virtual joystick influences the direction of the robot. The operator can view the environment via a live camera feed, which is displayed as an overlay. In the experiment, the participant has to navigate the robot from a control room to a specified destination using all three control methods. The participant has no direct visual contact with the robot and can therefore only orient himself using the video stream from the robot camera. The results show that the effectiveness of the VR control mode is the same as for the MR finger gestures mode. However, the VR Control mode performed better than the other two methods in terms of crashes. There were also difficulties in handling the four keys with the traditional keyboard method, which led to confusion in stressful situations, particularly.

The study showed that even inexperienced MR and VR users performed very well with the immersive methods. It also showed that the combination of robotics and immersive technologies offers considerable advantages over conventional control techniques. According to the paper, the combination of hand tracking technology and VR headsets represents a meaningful benefit. It is also assumed that VR headsets with embedded hand movement technology will become a mainstream alternative in the future.



(a) Hand gestures in Mixed Reality



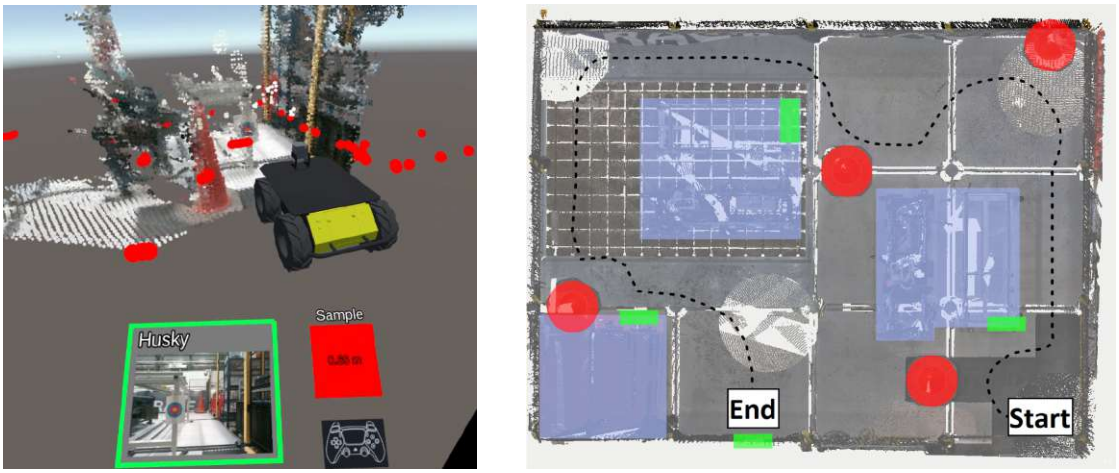
(b) Joystick mode in Virtual Reality

Figure 2.5: Robot control methods by Batistute et al. [2].

Stedman et al. [7] present an interface for remote control of a robot with dense 3D reconstruction and video stream. In a user study, a VR interface is compared with a traditional interface in a simulated nuclear monitoring task.

The results showed that users with the VR interface took longer to complete the task. However, these users had fewer collisions and rated the 3D map as more important than users with the traditional interface. It was also shown that the users with VR had a reduced cognitive workload during the task. However, they experienced a higher physical workload during the experiment. The study has shown that interfaces with VR may be suitable for the teleoperation of robots in the nuclear industry. Figure 2.6a shows the VR teleoperation interface in action, and Figure 2.6b shows the experiment course.

Walker et al. [26] developed an Augmented Reality interface that helps users to directly understand how their actions influence a robot through direct feedback. For this purpose,



(a) VR teleoperation interface (b) Experiment course

Figure 2.6: Visualizations from work by Stedman et al. [7].

they used immersive virtual surrogate robots (see Figure 2.7) and tested them on an aerial robot.

They compared the concepts of real-time virtual surrogates to waypoint virtual surrogates:

- *Real-time virtual surrogate*: With the concept of real-time virtual surrogates, the physical robot is not controlled directly. Instead, the user's commands are forwarded to a virtual surrogate, where the surrogates are then used as the basis for a planning algorithm. This algorithm runs on the physical robot and forces the robot to continuously track the surrogate. The physical robot stops when it reaches the same position as the virtual surrogate.

- *Waypoint virtual surrogate*: The waypoint virtual surrogate is an extension of the real-time virtual surrogate provided with improved support for planning the way ahead. The operator can trigger the robot to start the execution of the planned path. The path is defined by target points in the form of waypoints. It is possible for the operator to change the position of the waypoint, delete waypoints, and add new waypoints while the robot is on its way.

Besides these two methods, direct control of the physical robot was also implemented. Walker et al. [26] used the direct control as a baseline teleoperation system, where users navigated the physical directly instead of steering a virtual surrogate. The findings of the study indicated that users were faster at completing tasks with the surrogate designs than with the baseline teleoperation system.

A simple method of visualizing the robot's surroundings for teleoperating is to use video streams. One concept is the direct HMD teleoperation interface,e which provides the

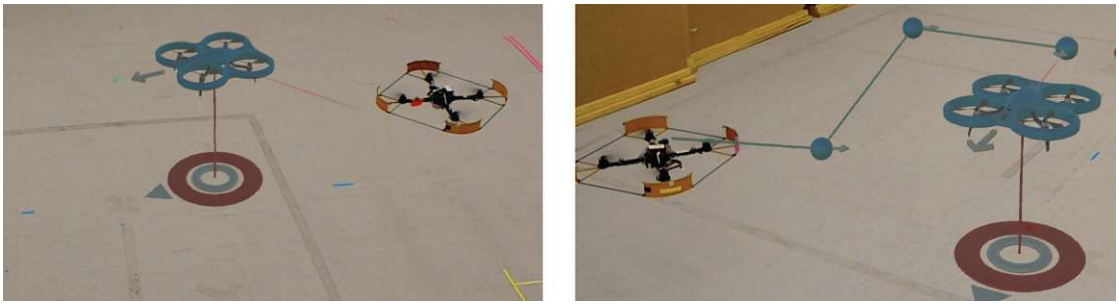


Figure 2.7: Virtual robot surrogate design for teleoperation in augmented reality by Walker et al. [26]. Left: Real-time virtual surrogate design. Right: Waypoint virtual surrogate system.

user with a live stream from the robot's perspective. This involves transmitting the robot's stereo video stream live to the HMD's lenses. This makes it possible to see the surroundings directly through the robot's point of view. Especially when the movement of the head directly controls the movement of the robot, as is the case with HMD head tracking, the teleoperation task can be improved. However, there is also a significant disadvantage of this method if the movement of the remote robot and the user do not match, resulting in contradictory signals. For example, if the robot moves but the user does not, this can lead to nausea. The solution to this issue is to decouple the user's eyes from the robot system by using AV HMD teleoperation interfaces that prevent conflicts in perception. These paradigms are called *Virtual Control Room* and *Cyber-Physical Interfaces*. They place the user in an augmented virtuality environment where the user's eyes are represented by a virtual camera. This decoupling reduces the nausea caused by the delay between the user's head movement and the movement of the robot. Summarizing, it is safe to say that using a VR interface with HMD as display hardware offers a more immersive visualization than traditional 2D displays [35, 36].

In their paper, Walker et al. [25] discuss the development and assessment of an immersive mixed reality interface for controlling a mobile robot in a human-robot team. The interface, presented as a *Cyber-Physical Control Room*, provides the user with a live 3D video stream and a live dense 3D point cloud for orientation in the environment. By combining two views, it is possible to provide both a robot-egocentric and a robot-exocentric perspective. For their immersive interfaces, they use a LiDAR (Light Detection and Ranging) sensor to create a 360° room-sized 3D reconstruction in the form of a dense point cloud. In the proposed interface, they combine the point cloud with a live video stream to provide both a robot-egocentric and a robot-exocentric perspective (see Figure 2.8). In a field experiment, the immersive interface is compared with a traditional control method. The evaluation shows that the immersive interface improves the effectiveness of navigation when performing a visual search in a complex environment by 28%. The Cyber-Physical Control Room improves human-robot teaming, for example, social engagement. The point cloud provides a good overview of the environment, but does not offer a detailed representation of the surroundings.

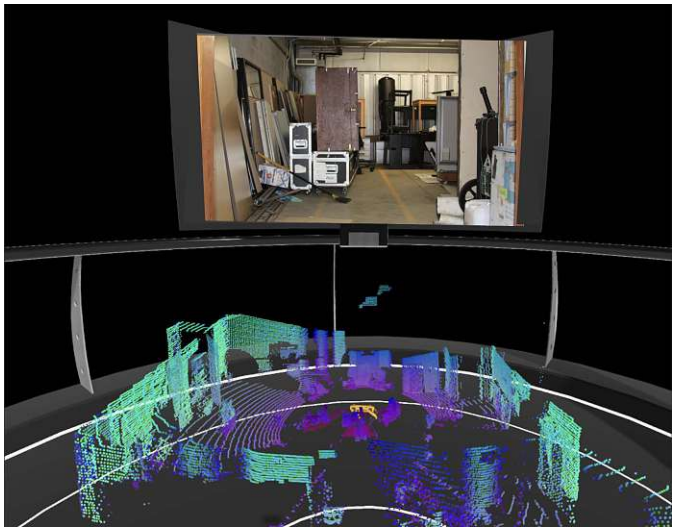


Figure 2.8: Immersive Cyber-Physical Control Room interface with live 3D video streams and 360° 3D point cloud within a virtual environment by Walker et al. [25].

In contrast, 3D meshes provide a realistic representation of the environment due to their faces and colors. This also leads to an intuitive interpretation of the scene. In the next section 3D reconstruction and 3D meshes are discussed in more detail.

2.2.1 3D Reconstruction

When teleoperating a robot, the representation of the environment for the operator is of crucial importance. The basis for this is always one or more sensors that capture the environment and are usually mounted on the robot. One simple method is to transfer camera images to the user in real-time. This enables the user to see the robot's surroundings from the camera's perspective. The presentation of the robot environment through a live camera feed is a simple method, which has also been used in combination with mixed reality and virtual reality systems to evaluate the efficiency of a robot's teleoperation [2].

However, the representation of the environment via camera images alone only provides a limited perspective, as the viewing angle is restricted to the camera's field of view. A purely video-based approach, therefore, suffers from limited situational awareness and a limited degree of immersion. The limited camera view also makes navigation and orientation in the environment more challenging [11].

3D reconstructions that create a three-dimensional model of the environment offer improvements. Three-dimensional representations of an environment can be created using various methods. A very common method is SLAM (Simultaneous Localization and Mapping), which addresses the problem of robot navigation in unknown environments. Specifically, the SLAM problem consists of creating a map of an unknown environment

for a robot moving in this environment and simultaneously determining the position of the robot relative to this map. SLAM is intended to enable the simultaneous localization and mapping of an unknown environment. Localization refers to the estimation of the robot's position, while mapping refers to the creation of the map. This allows a map to be created and the robot to be localized at the same time. This results in two interdependent challenges. Accurate localization requires an accurate map of the environment, but an accurate map also requires an exact position of the robot. Ensuring both at the same time is therefore a difficult task. To generate a 3D model with SLAM, sensors are required to provide the data for reconstruction [37, 38].

In their research on immersive robot teleoperation and scene exploration, Stotko et al. [11] use a SLAM-based system (see Figure 2.9) that captures live RGB-D data and creates a global 3D model from it. The captured scene elements are transmitted to a user who can observe the result via an HMD. In a conclusive user study, remote control via the SLAM-based system was compared with a purely video-based system. The robot's omnidirectional velocity is controlled via a wireless gamepad interface. It was shown that immersive robot teleoperation increases the level of situational awareness and enables more exact navigation in complex environments in contrast to purely video-based control.

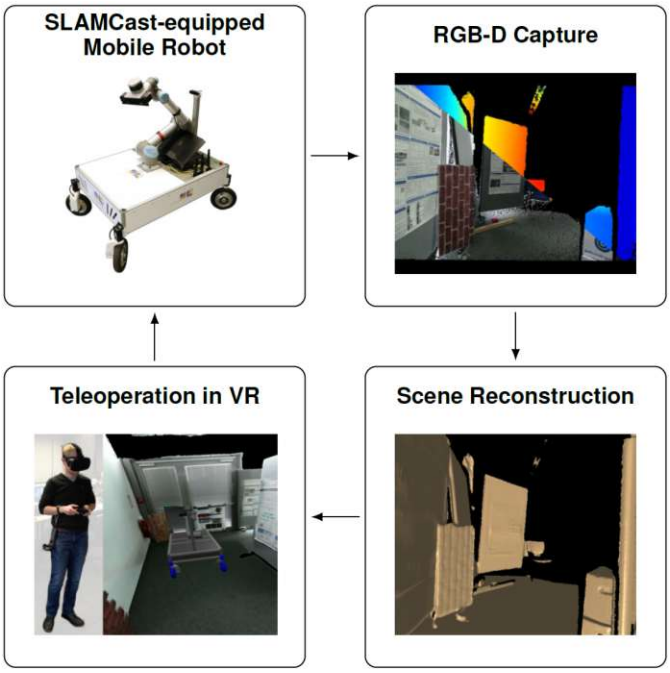


Figure 2.9: Overview of VR-based system for scene exploration and immersive robot teleoperation controlled by an operator [11].

CHAPTER 3

VIMREX - Virtual Interface for Mobile Robot Exploration

This chapter presents the design and development of a system for the remote control of a robot in virtual reality. Section 3.1 provides an overview of the hardware and software architecture and presents the pipeline for creating the virtual environment and controlling the robot. Section 3.2 shows how the camera images are captured and processed into a 3D model. The reconstruction pipeline is also described in more detail and the technology stack used is discussed. The integration of the created 3D model into Unity3D is covered in section 3.3. This section also deals with the navigation metaphors used and how they are used to control the physical robot. Section 3.4 provides a more detailed explanation of the network structure and its challenges. Section 3.5 explains how and what data is stored by the user study. The final section presents an additionally developed tool for later analysis of the recorded user study data.

3.1 Hardware and Software Architecture

To ensure the control of a robot by the user, a system is required which is divided into hardware components (see Figure 3.1) and software components. The hardware consists of a Boston Dynamics Spot robot on which a Spot Core payload is mounted. An external SSD (solid-state drive), a Wi-Fi USB dongle and a depth camera are also mounted on the Spot robot and connected to the Spot Core.
The Spot Core is connected to the Spot robot for the power supply. Spot Core is a hardware module for the Boston Dynamics Spot robot, providing additional computing capability, network and data interfaces [39]. A desktop computer and a HTC VIVE Pro Eye headset [1] with two VR controllers are used to reconstruct and investigate the

[1] https://www.vive.com/sea/product/vive-pro-eye/overview/

environment and control the robot. Ubuntu 22.04 LTS was installed on the external SSD connected to the Spot Core in order to be able to transfer the camera data wireless to the desktop computer. A Wi-Fi router ensures the wireless transmission of data between the devices.

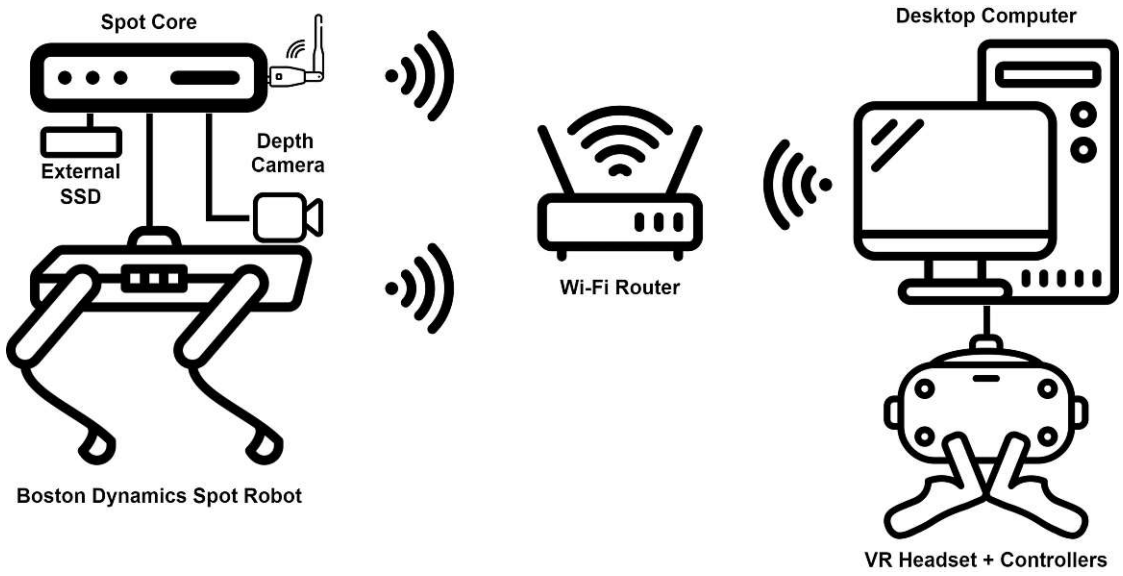


Figure 3.1: Hardware setup used in the thesis, featuring the Spot robot and the connected equipment, Wi-Fi router, desktop computer and VR headset with controllers [40].

All data exchange between the devices takes place via the TCP protocol. On the software side, the camera frames are sent from the Spot Core to the desktop computer via a TCP connection using the Intel RealSense SDK in a Python script. In the desktop computer, a 3D mesh is created from the input images using the Dense SLAM method of the Open3D [2] library. The created 3D mesh is then transferred internally in the desktop computer to the Unity3D application, in where the user can view it via a VR headset.
The camera pose data, which is used to position the virtual robot model in the virtual environment, is also transferred via a TCP connection. The movement data is calculated in Unity3D to control the physical robot. This data is sent to a Python script via a TCP connection, where it is then transferred to the Spot robot using the Spot SDK, which then executes the movements. As can be seen in the pipeline (see Figure 3.2), the movement of the physical robot results in a new position in real space, whereby frames are recorded from a new position in the environment and the reconstruction process receives new input data.

[2] https://www.open3d.org/

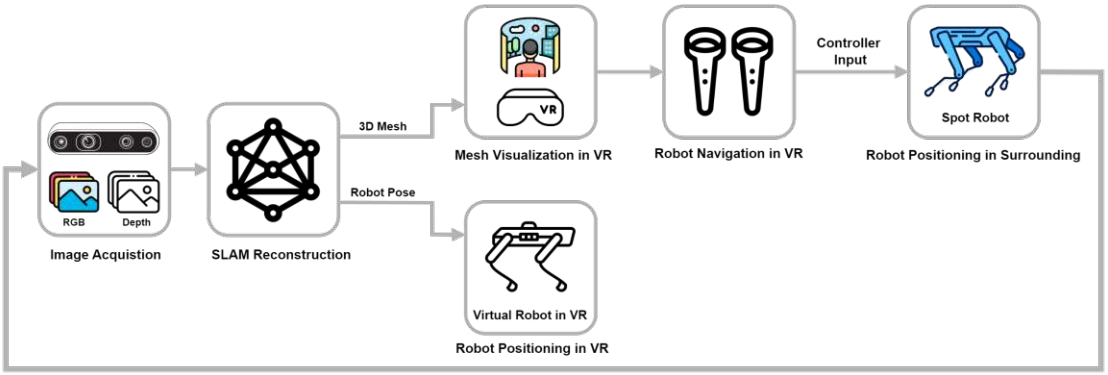


Figure 3.2: Overview of the 3D mesh reconstruction process, virtual robot model positioning and physical robot navigation [40].

3.2 3D Reconstruction

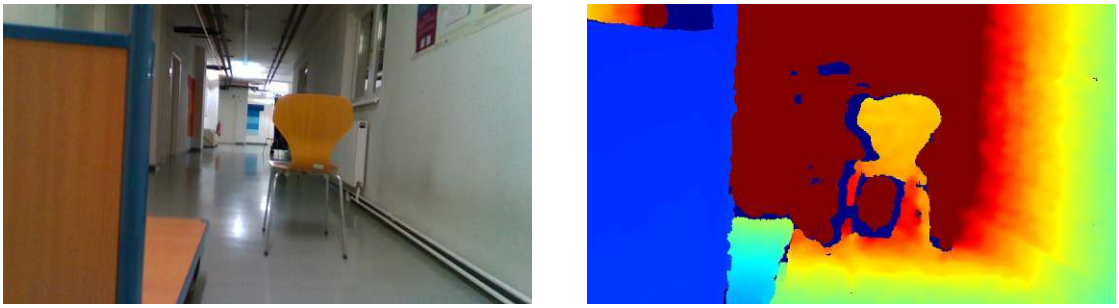
This section deals with the 3D reconstruction of the environment. The section begins with the acquisition of the image data from an RGB-D camera and then continues with the reconstruction pipeline, which is responsible for creating the 3D model.

3.2.1 Image Data Acquisition

In the field of computer vision and 3D reconstruction, RGB-D cameras have proven to be a major innovation. They allow the recording of visual information in three dimensions by combining RGB color data with depth data for each pixel (see Figure 3.3). RGB-D cameras can create a comprehensive depth map of the observed scene, which opens up many application possibilities. These cameras are also characterized by their cost efficiency, compact size and low cost [41].
Due to these advantages, the use of an RGB-D camera for recording the image data was the obvious choice. Specifically, the Intel RealSense Depth Camera D435i [3] (see Figure 3.4) was used, which has an RGB sensor as well as a stereo vision depth camera.

This camera model includes an inertial measurement unit (IMU), though it is not utilized in this thesis. The depth reconstruction benefits from active lighting and is achieved by evaluating the differences between two RGB images captured by two distinct sensors, enhancing the texture quality of the environment [41]. For a good balance between quality and data transfer size, a resolution of 848 x 480 pixels at 30 fps was used. The camera was mounted to the front part of the robot (see Figure 3.5) so that it could look forward in a stable way. A specially created mount consisting of a platform, which was fixed to the payload mount points with two screws, and the upper part of a camera tripod served as a tripod for the camera.

[3] https://www.intelrealsense.com/depth-camera-d435i/



(a) RGB frame (b) Colorized depth frame

Figure 3.3: Exemplary representation of a scene from the perspective of the depth camera visualized by a RGB frame and a depth frame.

Figure 3.4: Intel RealSense Depth Camera D435i [42]

The Spot Core was used to record, process, and transmit the image data. This is an accessory for the Spot robot that provides additional computing power and is supplied with power via the Spot's payload port. The input of the image data was implemented with a script written in Python, which also uses the Intel RealSense SDK 2.0 library [4] to access the sensors. The script also contains a TCP server via which the images are transmitted. First, the camera was configured, for which the resolution, format, and FPS were selected for the color images and depth images, respectively. Then intrinsic camera parameters (width, height, intrinsic matrix) were extracted from the first frame and saved. The saved values were then transferred to the desktop computer for the 3D reconstruction. Now, depending on the frames per second set for reading out the frames, the color and depth images are retrieved. The color and depth images are then aligned with each other. The alignment is important due to the different perspectives of the color and depth images. This also ensures that the pixels of both images match exactly and that the depth information is correctly linked to the corresponding pixel in the color image. Before the images are transferred, the color image is compressed in JPEG format to the configured JPEG quality. This enables a smaller file size and, therefore, faster transmission in the network. Both images are then converted into a byte object and transmitted to the connected client via a TCP connection.

[4] https://github.com/IntelRealSense/librealsense

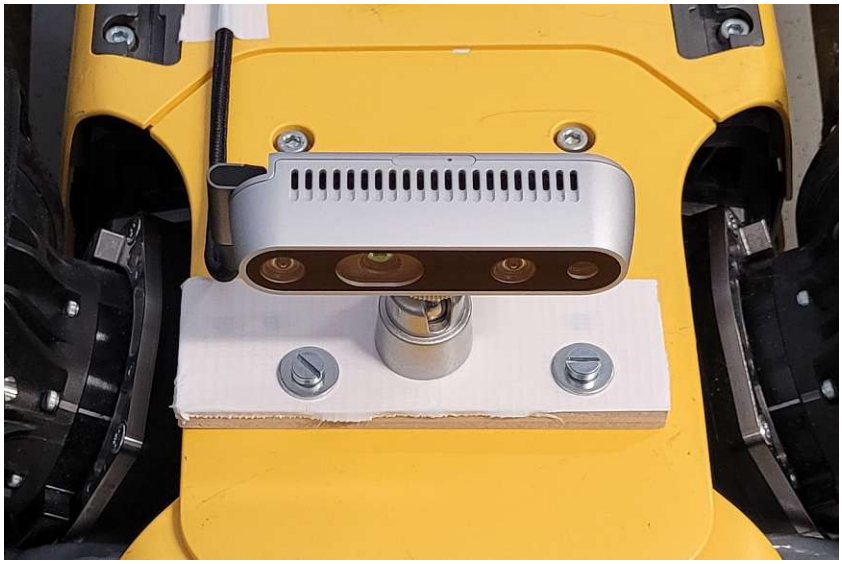


Figure 3.5: Depth camera mounted on Spot robot.

3.2.2 Reconstruction Pipeline

The TCP client for receiving the image data is executed on a separate computer, which is referred to as a 'desktop computer' in the context of this work. The images are then received via a Python script and passed to the pipeline for reconstruction in the same script.

The reconstruction is carried out using the open-source Open3D library, which works with Dense SLAM to assemble the received images into a virtual scene. Open3D is characterized by the two design principles of usefulness and ease of use. This is reflected in the fact that it supports common representations, algorithms and platforms. This includes support for the data structures: point clouds, meshes and RGB-D images. For all three of them complete sets of basic processing algorithms have been implemented [43]. Using Python, Open3D supports GPU acceleration through CUDA only on Linux [44]. GPU acceleration is necessary for a significantly faster reconstruction. However, as the Windows platform was required for the implementation of the VR application, the reconstruction and the VR application had to be carried out on two different platforms (Windows and Linux). In order to be able to perform both tasks on the same device, it was decided after some tests to use Windows Subsystem for Linux 2 (WSL2) [5]. WSL2 makes it possible to install a Linux distribution directly under Windows and run Linux applications. The distribution can access the system's hardware directly, allowing it to benefit from GPU acceleration via CUDA.

The code used for the 3D reconstruction originates from an example of the Open3D library, in which a dense RGB-D SLAM system is provided based on the fast volumetric

[5] https://learn.microsoft.com/de-de/windows/wsl/install

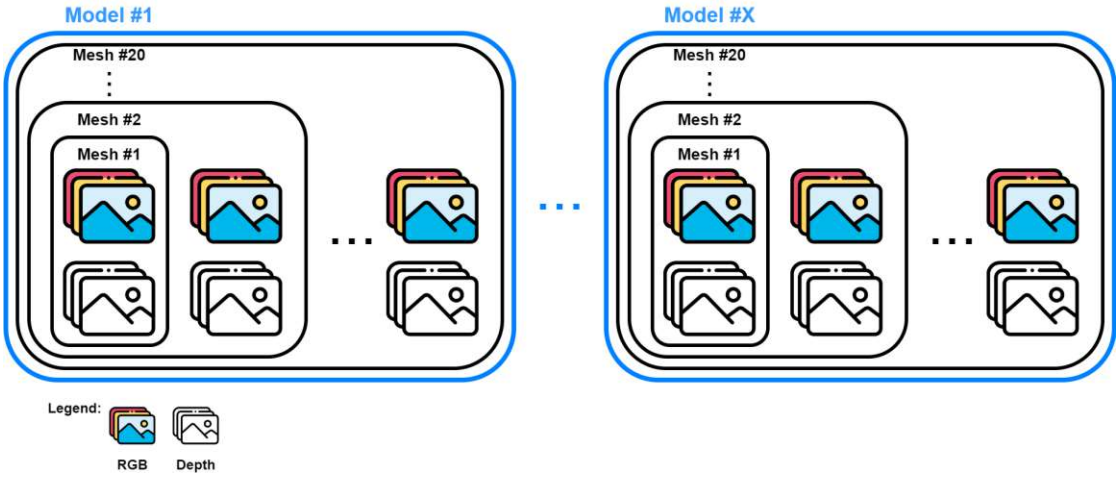


Figure 3.6: Overview of the procedure for integrating images into a model and their limited number of meshes per model [40].

reconstruction backend [45]. This example has a GUI that makes it possible to configure the tensor-based reconstruction [46] and see live how the scene is continuously assembled. The reconstruction system, which works with tensors, benefits greatly from the high-performance graphics card, which has 24 GB of graphics memory. The dense RGB-D SLAM system with frame-to-model tracking is powered by a fast volumetric reconstruction backend. The system uses a model based on a Voxel Block Grid that creates a synthetic frame from the model using volumetric ray casting. At the beginning of the pipeline, a volumetric model is created for dense SLAM, which is then gradually integrated with new images. Synthesizing the frame from the volumetric model using ray casting also helps to calculate the correct position of the camera in the scene. The tensor version of RGB-D Odometry is used for frame-to-model tracking [45].

For the purpose of this work, the original Python script had to be extended with additional functionalities. As it is not necessary to display the reconstruction in a GUI, all elements relating to the graphical interface were removed. In the original system, the frames are read from a folder. In order to receive the frames from Spot Core and process them directly, a TCP client function was added to the script. The client connects to the server running on the Spot Core and receives the color and depth frames. In order to ensure real-time visualization of the environment, it is necessary to make new sections of the reconstructed mesh available to the user at short intervals. For this purpose, a triangle mesh is extracted from the model at fixed intervals.

As the large amount of input data has led to very high GPU memory consumption during reconstruction, it was decided to create a new model after a certain number of meshes had been sent. This significantly reduces GPU memory consumption. In each iteration, 20 new frames (20 color frames and 20 depth frames) are added to the model. As soon as the frames are integrated into the model, a mesh is extracted from it, which can then

be further processed or sent. In this way, a virtual representation of the environment is gradually created. When 20 meshes (see Figure 3.6) have been extracted from the model and sent, a new model is created, and the old one is deleted. This measure significantly reduces the GPU memory requirement. Especially when the system is in use for a long time, the increasing size of the model would completely fill the GPU memory and thus slow down the system considerably or lead to a system crash.

Plane Segmentation

The use of multiple models with Dense SLAM in Open3D means that the floor surfaces of some models are not aligned exactly horizontally. This can be caused by inaccuracies in the SLAM process. In order to obtain a visually accurate scene, the floor is therefore aligned using plane segmentation.

The plane segmentation [47] feature is already available in Open3D, which is why the existing solution was used for integration into the system. The correction is always carried out shortly before a new model is created. Specifically, this means that shortly before the 20th mesh, and therefore the last mesh is extracted from the model, the inclination of the floor is checked so that it can be corrected if necessary. This means that the model contains as much information as possible about the scene in order to be able to recognize the ground. It starts by extracting a point cloud from the model, which can then be further processed. The aim is a planar segmentation to obtain a horizontal plane in the point cloud.
The original version of Plane Segmentation selects random groups of points and calculates a plane for each group. The plane with the most inliers is considered the best estimate and is returned. However, it is possible that a wall is selected instead of the floor. Figure 3.7 shows an example of a point cloud where the inliers are marked in red, making it clear which points form the plane. The outliers retain their original color.

To do this, it is necessary to find multiple planes and then decide on the basis of the equation whether there is a high probability that it is a floor. The *Multiple Planes Detection* [6] repository offers a simple and fast method for detecting multiple planes from a point cloud with RANSAC. The code was modified so that only planes that correspond to the floor of the point cloud with a high probability are selected. As a result, the method returns a plane equation (see Equation 3.1). The parameters a,b and c provide the orientation of the plane in space as a normal vector. Figure 3.8 shows the plane determined as the floor of the point cloud, with only the inlier points shown.

General form of plane equation:

$$ax + by + cz + d = 0 \tag{3.1}$$

To align the floor horizontally, the angle between the plane and a perfectly horizontal surface in space is calculated using the parameters (a, b, c). The angles are then used to

[6] https://github.com/yuecideng/Multiple_Planes_Detection

Figure 3.7: Visualization of Plane Segmentation results. Inliers, points belonging to the detected plane, are shown in red. Outliers, not part of the plane, retain their original color.

determine two rotation matrices for the rotation around the x and z axes. The matrix multiplication of the two rotation matrices results in a new rotation matrix, which makes it possible to rotate the plane around both axes. The resulting rotation matrix (see Equation 3.2) is used as input for an Open3D function to rotate the triangle mesh, which corrects the original inclination of the ground. The correction is intended to improve the visual quality of the reconstruction and create a seamless transition between the 3D models in the virtual scene. Regardless of whether it is the last mesh of a model and thus the procedure for correcting the inclination has been carried out, each mesh is still shifted by the height of the robot camera. The height of the real robot camera above the ground was measured manually and its correction is intended to ensure that the virtual

Figure 3.8: Multiple Planes Detection result. The detected floor area and its corresponding points on the plane are highlighted in color.

robot model is always at the correct height. Each mesh is then added to a data queue so that it can be accessed by other functions.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.2}$$

A separate thread then deals with sending the mesh via a TCP connection. First, however, the data is reduced and serialized in a separate function. This serves to extract only necessary information from the given mesh, which is optimized for network transmission. First, the triangle indices, vertex positions and color values are extracted from the mesh. As not all vertices are required for the surfaces, only the actual vertices referenced in the triangles are taken into account.
Therefore, only the corresponding colors are selected. The serialization of the data begins with the determination of the total size of the message, which is made up of the sizes of the respective arrays and the headers (16 bytes).

variable	message size	vertices length	vertices data	triangles length	triangles data	color length	color data
size in bytes	4 bytes	4 bytes	N bytes	4 bytes	N bytes	4 bytes	N bytes

Table 3.1: Structure of the mesh byte array used for transferring the mesh to the Unity application.

The resulting byte array is then a compact representation of the mesh, which can be easily deserialized later using the embedded headers. This byte array is then transferred to Unity3D via a TCP connection.

3.3 VR Interface

This section discusses the integration of the 3D model into the virtual scene and deals with the positioning of the virtual robot model. Furthermore, the method for controlling the user view in Unity is explained. The implementation of two navigation metaphors for the remote control of the robot is shown in this section. Finally, it is explained how the physical robot is controlled in the real environment based on the metaphors presented.

3.3.1 Mesh Integration in Virtual Reality

The virtual reality application was developed with the game engine Unity3D using the OpenXR [7] plugin. OpenXR enables the uncomplicated development of AR/VR applications for numerous AR/VR devices. The use of the `XR Interaction Toolkit` [8] package supports the development of user interactions in the virtual environment.

The meshes are received in Unity3D via a script that contains a TCP socket. The data is first stored in a byte array. The byte data is then converted into a Vector3 array for the vertices, an Int array for the triangles and a Color array for the colors. A mesh is then created in Unity3D, to which the respective arrays for vertices, triangles and colors are added. The MeshFilter and MeshRenderer components are then added to a newly created GameObject. Then the previously created mesh is transferred to the MeshFilter. A newly added mesh that originates from the same model as the previous one always contains redundant information, as it has only been extended with new sections. This is why Unity3D deletes the previous mesh each time a new one is added. Only the last mesh from a model before a new model is created is retained. It is possible for the user to view the generated mesh via their virtual reality headset. By rotating the head, it is possible to obtain a comprehensive view of the surroundings. A virtual robot (see Figure

[7] `https://docs.Unity3D3d.com/Packages/com.Unity3D.xr.openxr@0.1/manual/index.html`

[8] `https://docs.Unity3D3d.com/Packages/com.Unity3D.xr.interaction.toolkit@3.0/manual/index.html`

3.10) in the scene is used to illustrate the current position of the physical robot in the environment. The next section discusses the exact positioning of the virtual robot in more detail.

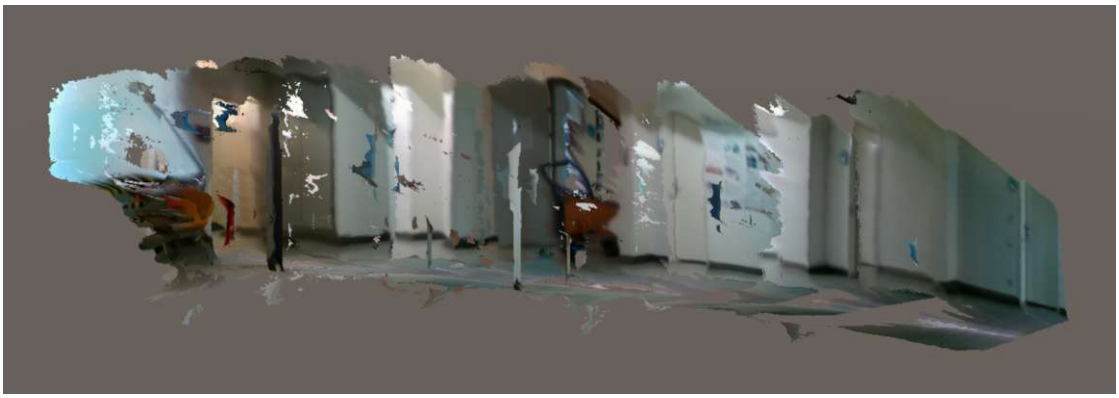


Figure 3.9: Several meshes in Unity building the virtual representation of the real environment.

3.3.2 Virtual Robot Pose Pipeline

Another important aspect of the SLAM method is the determination of the camera position within the reconstructed scene. This feature is used to obtain the position of the camera for the virtual robot in the virtual environment. In the reconstruction script, the frame is continuously tracked in relation to the model and the result is saved in a transformation matrix. This transformation matrix is then used to obtain the current camera position and camera rotation at defined time intervals. To obtain the current rotation of the camera, the rotation matrix is extracted from the transformation matrix. Quaternion angles are then used to transfer the rotations. Quaternion is a vector of four tuples that can describe any rotation in a three-dimensional space. The advantages of quaternion lie in the continuous and unambiguous representation of rotations in three-dimensional space. This also avoids the gimbal lock problem. The gimbal lock problem occurs when Euler angles are used, whereby two of the three rotation axes coincide. This leads to the loss of one degree of freedom in the rotation, which is particularly problematic with complex movements [48].
Since Unity3D also works internally with quaternions to display rotations, it makes a lot of sense to transfer these values directly. The position and rotation data is then saved as a JSON string and added to a queue.

The data is then processed asynchronously in another thread, read from the queue and transferred via a TCP connection. In the Unity3D application there is then a TCP socket which receives the data, converts it into a suitable format and transfers it to the virtual robot (see Figure 3.10) as a new position and rotation. For a smooth movement between the current values and the target values, linear interpolation is used for the position data and spherical linear interpolation for the rotation data.

Figure 3.10: Virtual robot model in the virtual environment in Unity3D.

3.3.3 User Navigation in Virtual Reality

The user should also be able to move through the environment. Physical movement within the play area is only minimally feasible due to the limited size of the test room and the wired connection of the VR headset to the computer. Therefore the functionality of teleportation was selected for moving around and following the robot in the virtual environment. This allows the user to select the next point to which the user wants to teleport using the trigger button on the blue controller (see Figure 3.11). A ray appears from the controller which has a certain curve shape, which then intersects with the ground. The position at which the ray and the ground intersect then appears as a teleport reticle. This reticle then also marks the position to which the user is teleported when pressing the trigger button and thus confirms the teleportation. The existing functions of the XR Interaction Toolkit, which includes the Teleportation Provider and Teleportation Area components, were used for this purpose. In combination with the XR Ray Interactor, Line Renderer and XR Interactor Line Visual components, teleportation can be easily implemented via the controller.

3.3.4 Physical Robot Teleoperation in Unity

This section deals with the teleoperation of the physical, mobile robot in the real environment and describes the implementation in Unity3D.

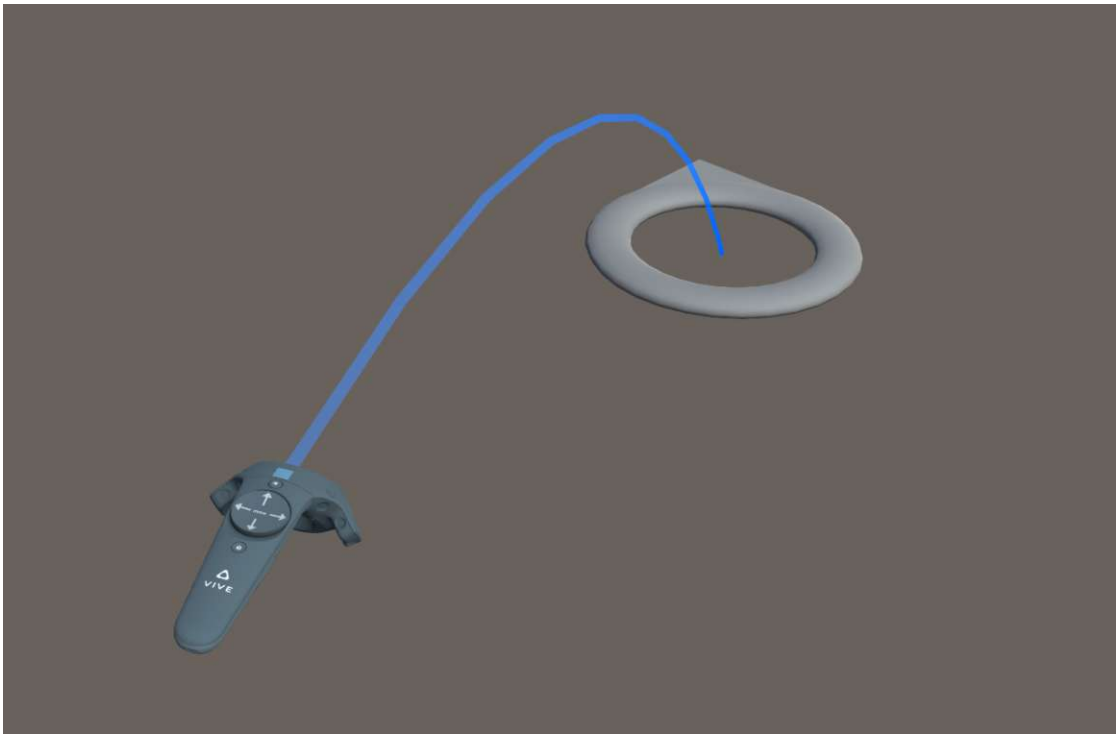


Figure 3.11: Illustration of user teleportation method in virtual reality. A ray emits from the controller which intersects with the floor and teleports the user to the targeted point.

For the user study, two navigation metaphors are to be compared: *Direct Control* and *Point-and-Click*. The implementation and the usage of the two metaphors is described in this section. Two controllers are used for this purpose, which are color-coded in the virtual scene (see Figure 3.12). The color coding serves to improve communication between the study supervisor and the participants so that fewer misunderstandings can arise when explaining the functionality.

Direct Control

This navigation metaphor is based on classic methods of controlling an object, using gamepad-like navigation to move and rotate an element. Two VR controllers are required for the implementation. In this study a HTC VIVE Pro Eye headset is used and therefore the controlling is done by the HTC VIVE controllers. These have a trackpad on both controllers, which has two axes (value between -1 and 1) for input and also reacts to touch inputs. The controller marked in blue controls the movement forwards, backwards, right and left. The controller marked in red controls the rotation via the horizontal axis (see Figure 3.12). The input data is queried at a defined time interval and converted into JSON.

Figure 3.12: Virtual controllers for direct control of the mobile robot. Left controller: handles linear movement to move forward, backward and to the left and right. Right controller: responsible for rotational motion to turn left and right.

Point-and-Click

With this metaphor, the robot is not controlled directly via the input data, but a target point is set for the robot, which it then has to reach. This method uses some components that are also used for teleportation, such as the Ray Interactor and the Line Renderer component. The controller marked in red is used for this method. If this control mode has been selected in the user study, a ray also appears from the controller (see Figure 3.13). If the ray intersects with the ground, a transparent robot model appears at this point, marking the possible waypoint. The position of the waypoint can be determined by moving the red-colored controller. Using the horizontal axis of the red-colored controller trackpad, the rotation of the transparent robot model can be changed. By confirming the placing process via the trigger button of the controller, a waypoint in the form of a robot model is set at this point. The difference between the virtual robot and the waypoint is then calculated in the code and converted into a JSON. The JSON string is then transferred from Unity to the Python script.

3.3.5 Physical Robot Control Pipeline

This section discusses the transfer of the robot movement data from Unity3D via the Python script to the real robot. This section follows on from the previous one, which explained how the control of each metaphor is implemented in Unity. The starting point for controlling the robot is the transmission of JSON data containing either only

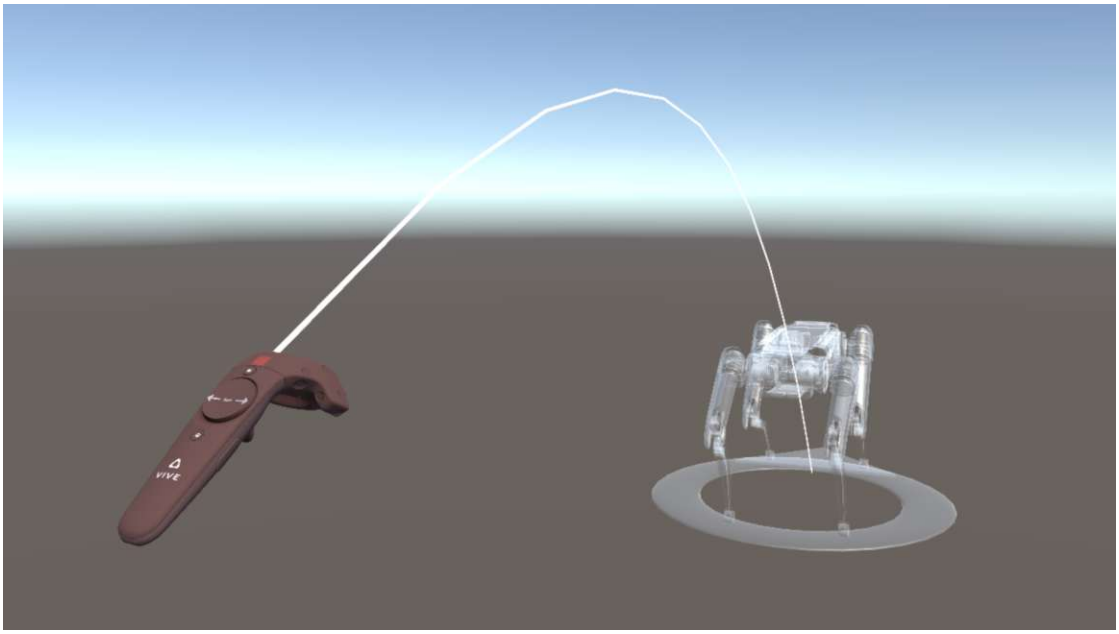


Figure 3.13: Visual representation of the *Point-and-Click* metaphor for placing a new waypoint. A ray emits from the controller which intersects with the floor and displays a transparent robot model, which serves as an aiming tool.

a command or a command with additional position and rotation information. The movement data is saved in a JSON object. The object contains various fields so that the robot knows what to do. One of these fields is called "command" and specifically defines which actions the robot should perform.

List of used commands:

- *boot*: starts the robot's motors, robot stands up
- *move*: used for *Direct Control* metaphor, input from the controller trackpads controls the movement
- *moveGoal*: used for *Point-and-Click* metaphor, movement of the robot to the next position
- *sit*: robot sits on the floor
- *return*: robot moves back to the position at which the robot was booted
- *position*: returns the current position of the robot in the room
- *safeShutdown*: robot sits down on the floor before the robot is shut down
- *stop*: robot is shut down, stops the background thread

To start the robot, a JSON object with the command "boot" has to be sent. As a result, control of the robot is taken over and the motors are started. At the end of the boot process, the robot rises and remains in this position until a new command is received. When the robot is controlled using both navigation methods, a JSON object with the following fields is generated: command, x, y, yaw.

The JSON data is then transferred to a server via a TCP connection, which is implemented in a Python script. The use of a Python script as an intermediary results from the existence of the Spot SDK [9], which greatly simplifies communication using Python. The server implemented in Python receives the JSON data and first checks which command has arrived. It then decides how to proceed.

Direct Control

With *Direct Control* metaphor, the speed of the robot is determined using the input from the controller (value between -1 and +1) and the predefined speed constant `VELOCITY_BASE_SPEED`=0.2m/s. This calculation results in movement values for the robot along a 2D plane.

A Python script, that serves as an interface between the Unity3D application and the Spot robot, receives the JSON data. During direct control, the Unity application sends a "move" command, along with values for the x and y directions and the yaw rotation. These values are passed to a method of the Spot SDK, which generates short-term speed commands based on the received instructions. The control command with the specific speeds is then sent to the robot at a fixed time interval. This approach enables responsive control of robot movement in real time, which is suitable for manual navigation with VR controllers or gamepads.

Point-and-Click

If the robot is controlled using the point-and-click metaphor, both the desired distance and the rotation are specified. The Python script uses these transferred values to calculate the robot's target position. The Unity application sends a JSON string with the command "moveGoal", which contains the local position and rotation.

A trajectory-based movement control system moves the physical robot to the specified pose on a 2D plane. The target pose is determined within the body coordinate system and transformed into the global coordinate frame. If the target position has been calculated, the movement command is transmitted to the robot. The speed of the robot is limited by the speed constant (`VELOCITY_BASE_SPEED`=0.2m/s), which serves as an upper limit. While the robot executes the movement, the system continuously checks the progress in a loop and terminates the process if the robot reaches the target or an error occurs.

[9] `https://dev.bostondynamics.com/`

3.4 Networking

A wireless network connection is required to ensure the transmission of image data from the depth camera and the control of the robot. It was important to provide both a stable and fast connection in order to enable a delay-free display of the reconstructed environment and responsive control. The use of the access point provided by the Spot robot was initially considered as part of the familiarization with the operation of the Spot robot. The use of the access point would not require any additional hardware and is based on existing features of the robot. However, initial tests showed limitations in terms of bandwidth, which meant that a different solution was required. The robot also offers the option of connecting to another wireless network as a client. As part of a documentary experiment, the data transfer rate between the Spot Core and the desktop computer was compared.

3.4.1 Network Bandwidth Test

This test compares the data transmission speed between the Spot Core and the desktop computer under WSL2 via two network configurations. It compares the connection via the Spot Wi-Fi access point on the one hand and the connection via an external Wi-Fi router on the other.

The data rate is determined using two test methods:

iPerf3: The iPerf3 [10] tool is used to measure the maximum achievable bandwidth between Spot Core and WSL2. For this purpose, iPerf3 was installed on both devices. A server was started on the Spot Core, while the client was executed in the WSL2 environment. The test runs for 10 seconds by default and provides a brief summary of the measured bandwidth.

Image Transfer: To simulate a test scenario similar to the actual purpose, a previously recorded dataset was used. The dataset consisted of 4,312 images divided into 2,156 color images and 2,156 depth images. The images were captured with a resolution of 848x480 pixels and a frame rate of 30 frames per second. The use of a pre-recorded dataset simplifies the execution and reproducibility of the test. Two Python scripts were implemented, one acting as a server on the Spot Core and the other as a client in WSL2. After a successful connection, the test started and the image data was transferred. After all images were transferred, both scripts displayed statistics relating to the transfer.

Test Environment

The test was conducted in the VR Lab. The router was located outside the room with the door open, as shown in Figure 4.3. Both the Spot robot and the desktop computer were in the VR Lab during all test runs.

[10] https://iperf.fr/

Results

The results (see Table 3.2) show that the data transfer rate is significantly higher in both test scenarios using an external Wi-Fi router. Using the router, a transfer rate of ≈ 134.23 Mbps was measured for image transfer, in contrast to ≈ 22.42 Mbps with the Spot access point. Based on the statistics of the transmitted data, an average size of 248 kilobytes per RGB deep pair was determined. An RGB-depth pair consists of color data compressed into JPEG format and uncompressed depth data. If a target transmission rate of 30 RGB-depth pairs per second is to be achieved with a size of 248 kilobytes per RGB-depth pair, a minimum bandwidth of ≈ 59.52 Mbps must be available (see Equation 3.3).

$$248\,\text{KB} \times 30\,\text{pairs/s} = 7440\,\text{KBps} = 59,520\,\text{Kbps} = 59.52\,\text{Mbps} \tag{3.3}$$

This required bandwidth of 59.52 Mbps could not be achieved with 22.42 Mbps at the Spot access point, even at a short distance between the devices. Further tests with a greater distance between the router and robot confirmed these results and showed the performance advantage of the external router. For this reason, the Wi-Fi router was subsequently used for data transmission. The router also offers the advantage of being independent of the robot's power supply. In addition, the router can be permanently positioned in the room, which means that the distance between the desktop computer and the router remains constant. Thanks to the independent placement, the router can be positioned so that it is usually between the computer and robot for a consistent signal connection.

	Spot access point	**Wi-Fi Router**
iPerf3	≈ 18.85 Mbps	≈ 243.75 Mbps
Image transfer	≈ 22.42 Mbps	≈ 134.23 Mbps

Table 3.2: Overview of network bandwidth test results

3.4.2 Spot Network Configuration

To connect the robot to the Wi-Fi router, it is necessary to change the Wi-Fi Network Type from *Access Point* to *Client* in the Network Setup section of the Spot Admin Console. The Spot Core has a Wi-Fi USB dongle, which enables the connection to the Wi-Fi of the router. The desktop computer is also integrated into the network in order to receive the images from the Spot Core. The transfer of meshes between the reconstruction script in WSL2 and Unity3D is handled internally on the computer.

3.5 Study Data Export

In order to evaluate the user study, data were collected and stored. Particular focus was placed on the time that each participant needed to complete the task. For a later analysis

of the movements, the position and rotation data of the virtual robot and the VR headset in the virtual space were also saved for each navigation metaphor. In addition, the last mesh of a model that was displayed in the virtual scene was saved as an OBJ file. Colors were not used when saving the model. With this data, it is possible to recreate and visualize the participant's experience later on.

In a folder called "StudyLog", a new folder was created for each participant with the naming scheme "`User_<UserId>_<year>_<month>_<day>`". In this new folder, two CSV files per navigation metaphor were saved. One file includes the position/rotation of the virtual robot and the VR Headset at specific timestamps, and the other file stores mesh numbers at specific timestamps.

The structure of the CSV files is explained in the following:

- `position_<UserId>_<navigation metaphor>.csv`
  - *Timestamp*: current timestamp in the format "yyyy-MM-dd HH:mm:ss.ff"
  - *RobotPosition<coordinate>*: position of the robot, one column per coordinate
  - *RobotRotation<coordinate>*: rotation of the robot as a quaternion, one column per coordinate
  - *PlayerPosition<coordinate>*: position of the player, one column per coordinate
  - *PlayerRotation<coordinate>*: rotation of the player as quaternion, one column per coordinate
- `mesh_<UserId>_<navigation metaphor>.csv`
  - *Timestamp*: current timestamp in the format "yyyy-MM-dd HH:mm:ss.ff"
  - *MeshNumber*: number of the mesh that was created at this time

Another folder called "Mesh" is also created in the user specific folder and stores the meshes in the OBJ file format. The naming scheme is as follows: "`<navigation metaphor>_Mesh_<number of mesh>.obj`"

3.6 Trajectory Simulation

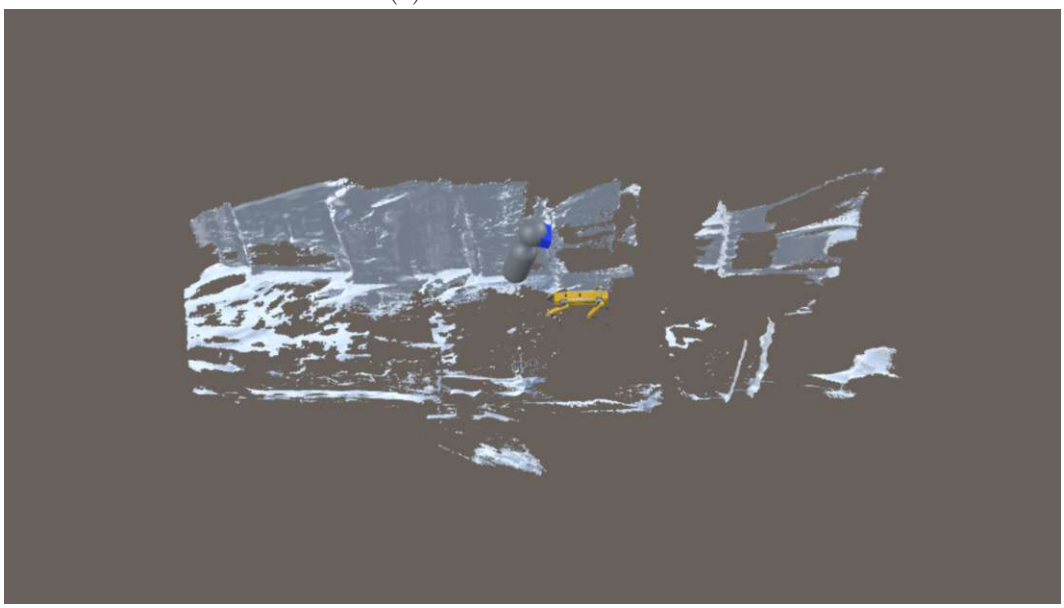
For the evaluation of the user study data, an application was developed with which it was possible to track the trajectories of the robot and participants. The application was developed in Unity3D, which makes it uncomplicated to handle both the CSV files and the mesh data. The development consisted of two parts, which can be executed via buttons in the Unity3D inspector. To start the trajectory simulation, the CSV files and the mesh data of a run have to be imported. In the inspector, the user can enter the name of the folder where the participant's data is stored. The navigation metaphor to be analyzed can be selected via a drop-down menu.

By clicking the "READ CSV" button in the inspector, the CSV file with the position and rotation data and the CSV file with the mesh data are imported.

Once the data has been read in, the user can start the simulation by clicking the “START SIMULATION” button. This starts by reading out the first entry of the pose data. Pose data includes the position and rotation of an object. The Robot Pose and Player Pose data are then passed to two GameObjects, which symbolize the player and virtual robot. Since the pose data was originally saved every second, new values are read from the list every second and passed to the objects.

The user is now able to track the trajectory of the player and the robot (see Figure 3.14). To ensure that the recorded environment is also included in the simulation, a saved mesh is loaded into the scene and displayed in the correct time sequence. The timestamp from the current entry of the CSV file with the position and rotation data is compared to the timestamp from the CSV file with the mesh data. When the timestamps correlate in a specified range, the mesh is imported into the visual scene. The application, therefore, enables a later evaluation of the user and robot interaction within the reconstructed environment.

(a) Scene with one mesh



(b) Scene with five meshes

Figure 3.14: Screenshots of the Trajectory Simulation application, which show the trajectory of the robot and user over time as well as the visualization of the environment.

CHAPTER 4

User Study

This chapter describes the evaluation of two different navigation metaphors for controlling a physical robot in VR based on a user study. A total of 12 participants (10 male, 2 female) aged between 25 and 34 were recruited for this study. First, the aim of the study is explained in more detail, followed by a description of the technical setup used to conduct the survey. The participants and the task are then described. This is followed by an overview of the data collected in the study and a introduction to the methods used to collect and analyze the data. Finally, the results are presented and interpreted in more detail.

4.1 Aim

The aim of the user study is to compare two navigation metaphors for the remote control of a mobile robot in virtual reality. In the study, the two navigation metaphors are to be evaluated in terms of their usability, intuitiveness and efficiency. During the experiment, the users are located in a test room separated from the environment in which the robot is to be controlled. Users are equipped with a VR headset and two VR controllers to interact with the robot. The real environment in which the physical robot moves is located in a hallway adjacent to the test room.
In two consecutive runs, the participants navigate the robot from a fixed starting position to a target point using one of the two navigation metaphors. Objective and subjective data is collected before the start of the experiment, between the runs and after completion of the tasks. The results of the user study should provide information on which navigation metaphors are more efficient, more intuitive and generally better qualified for the remote control of a mobile robot in VR. Finally this study should contribute to improving the interaction between humans and robots.

4.2 Technical Setup

The user study was conducted in the VR Lab at TU Wien and the adjacent hallway area. For the VR part of the study, an area measuring approximately 2.5 by 1.5 meters was available in the lab, in which the participant could move freely. This area functioned as the so-called "play area", within which the participant performed the interactions in virtual reality. The visual representation of the virtual environment was provided by an application developed with the Unity3D game engine. To ensure safety within the main play area, a boundary was displayed in the virtual environment as soon as the user moved closer to the boundaries of the play area. An HTC VIVE Pro Eye headset (see Figure 4.1c) was used for the study in combination with two HTC VIVE controllers. Tracking of the VR components was ensured by two fixed lighthouses installed in the VR Lab. The VR headset was connected to a desktop computer via a cable. The desktop computer was responsible for the 3D reconstruction, the visualization of the virtual environment in VR and the communication between the VR application in Unity3D and the robot. Technical specifications of the desktop computer are shown in Table 4.1.

The Boston Dynamics Spot (see Figure 4.1a) was used as the physical robot, on which an additional computer, the Spot Core, was mounted. Technical specifications of the Spot Core are displayed in Table 4.2. The Spot Core was connected via a cable to an external SSD on which Ubuntu 22.04 LTS was installed with all the necessary libraries and software packages. In addition, the Intel RealSense D435i depth camera, which was mounted on the front part of the robot, was connected to the Spot Core via a cable.

Manual control of the robot and monitoring of its position and status were done via a tablet, which was connected to the spot in combination with the corresponding app. The tablet was primarily used to manually navigate the robot to the starting position. The tablet can also be used to specify who currently controls the robot, as well as to release or take control. The additional monitoring of the situation via the robot cameras enables the study supervisor to ensure that the experiment runs safely. The robot basically prevents collisions with the environment by means of built-in safety mechanisms to avoid collisions. Nevertheless, it is essential that the study supervisor can intervene at any time. The study supervisor can use the tablet to take control of the robot in specific situations and control it manually at this point.

The network connection of the Spot Core was established via a Wi-Fi USB dongle. All wireless communication between all devices involved is handled via a Wi-Fi router (see Figure 4.1b) that all devices were connected to.

4.3 Participants

A total of 12 participants aged between 25 and 34 took part in the study. In terms of experience with virtual reality using a HMD, 2 participants reported that they had regular experience (level 6). 3 participants rated their experience as moderate (level 2-4), while 6 indicated only little experience (level 1). Two participants had no experience

OS	Windows 11
CPU	Intel i9-11900K (3.5GHz)
RAM	32 GB
GPU	NVIDIA RTX 3090

Table 4.1: Technical specifications of the desktop computer

OS	Ubuntu 22.04
CPU	Intel i5-8365UE
RAM	16 GB
GPU	Intel UHD Graphics 620

Table 4.2: Technical specifications of the Spot Core

with VR via HMD (level 0). Regarding the teleoperation with robots, 2 participants had little experience. The remaining 10 participants reported that they had no experience of teleoperation with robots.

4.4 Study Procedure

At the beginning of the study, the participants were informed about the aims, procedure, type and scope of the data collection using a consent form. Participants then completed a questionnaire in which their age, gender, experience with VR via HMD and experience with the teleoperation of robots were recorded. The participants were then shown the area in which they could move around during the experiment. After a brief introduction to the use of the VR headset and the VR controllers, the participants were allowed to put on the VR headset to familiarize themselves with VR.
Users were able to gain their first impressions and test the controllers in a virtual room provided via the SteamVR platform. Once the participants had familiarized themselves with the VR environment, the test setup was started. In order to familiarize themselves with both navigation metaphors (*Direct Control* and *Point-and-Click*), the robot was first placed in front of the VR Lab in the opposite direction to the later task.

The participants were able to try out both navigation metaphors one after the other to familiarize themselves with the control elements. Once the participants had experienced both control methods and felt confident with their handling, the first run of the study could begin. Before the start of a run, the physical robot was manually navigated to the starting position and placed on the floor. Since a possible training effect can occur if all participants start with the same navigation metaphor, the order of the navigation metaphor used was alternated for each participant (see Table 4.3).

Before the first run, the study supervisor informed the participants about the specific task and the goal they had to achieve. The aim of the task was to approach a large board with a checkerboard pattern at the end of the hallway with the robot and to stop approximately one meter away from it. This board was placed by the study supervisor at the same time as the participant completed the questionnaire. The participants were told that they had to complete this task within a time limit of 5 minutes.
Otherwise, the task was considered not completed. In order to make the VR experience and teleoperation as realistic as possible, the door between the VR lab and the hallway was usually closed during each run. This was to ensure that the participants could

(a) Boston Dynamics Spot robot equipped with Spot Core and a depth camera.

(b) Wi-Fi router used for wireless communication among the components.

(c) HTC VIVE Pro Eye headset and corresponding controllers for immersive user interaction within the virtual environment [49].

Figure 4.1: Overview of the key hardware components used in the user study.

UserID	Run 1	Run 2
101	Point-and-Click	Direct Control
102	Direct Control	Point-and-Click
103	Point-and-Click	Direct Control
104	Direct Control	Point-and-Click
105	Point-and-Click	Direct Control
106	Direct Control	Point-and-Click
107	Point-and-Click	Direct Control
108	Direct Control	Point-and-Click
109	Point-and-Click	Direct Control
110	Direct Control	Point-and-Click
111	Point-and-Click	Direct Control
112	Direct Control	Point-and-Click

Table 4.3: Table representing by which metaphor the user started into the experiment.

concentrate fully on controlling the robot within the virtual environment. When the participant was ready and the robot was correctly positioned in the hallway, the process of starting the system was initiated by the study supervisor.
The camera server was first started on the Spot Core, through which a client could connect and receive the depth and color frames. The server was then started, which was used to send the user's control commands to the robot.

The Unity3D application then started, enabling the user to see the virtual environment of the test scenario in the VR headset. A component within the Unity application then connected to the server to control the physical robot. This triggered the robot's boot process, which started the motors and raised the robot. Once the robot was standing upright, the script for 3D reconstruction of the environment from the camera images was started. This script communicated with the server on the Spot Core and received the images captured by the depth camera, generated a 3D mesh and sent it to the Unity application via TCP connection.
At the same time, the camera pose is transmitted to Unity at regular intervals. The received data was processed in Unity, the mesh was visualized in the virtual scene and the virtual robot model was positioned according to the transmitted camera pose.

After completing the task, the participant was asked to fill out a post-run questionnaire. While the participant was filling out the questionnaire and as long as there were no questions from the participant, the study supervisor placed the robot back at the starting position and checked the completeness of the recorded data.
After a short break, the second run was started, using the other navigation metaphor. Once this run had also been completed and the post-run questionnaire had been filled out again, a post-experiment questionnaire followed. Finally, participants had the option to provide general comments and remarks in an additional text field.

4.5 Task

The task in each run was to navigate the robot from the starting position to a checkerboard pattern within a time limit of 5 minutes and to stop it approximately one meter away from the pattern. To do this, it was necessary to control the robot using the selected navigation metaphor and to search for the pattern in the reconstructed environment. Specifically, the target with the checkerboard pattern was located at the end of a hallway in a small room where the board with the pattern was leaning against a door. The hallway is adjacent to the room in which the participant is located and is separated from it by a door. The distance between the starting point and the finish is approximately 22 meters, with the majority of the route extending along a straight hallway. Only about 1.5 meters before the end point was it necessary to navigate the robot slightly to the right through an opening the width of a standard door. This narrow section required a little more concentration and dexterity when controlling the robot. Figure 4.2 illustrates a user performing the task using the VR interface, the first-person VR perspective and the mobile robot in the hallway. An overview plan of the test environment with the starting point and target point noted can be found in Figure 4.3.

Figure 4.2: Image compilation of the user study execution. Left: User in the test room with the VR headset on and the VR controllers in hand. Middle: User view of robot and the *Point-and-Click* navigation metaphor in the reconstructed environment. Right: Physical robot in the hallway.

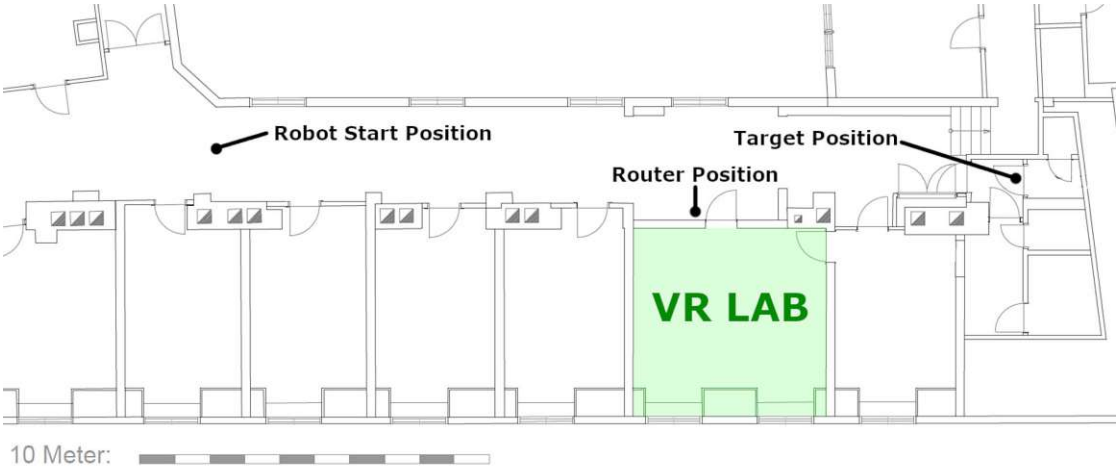


Figure 4.3: Representation of the test environment using a floor plan. The starting point of the robot in the user study and the target point are shown. The graphic shows the spatial relationship between the hallway and the VR Lab. Image based on [50].

4.6 Data Collection

Both objective and subjective data were collected in the user study. Objective data was collected during the run via functions in the Unity3D application such as task completion time. The results of the time measurements were part of the evaluation between the navigation metaphors.

Subjective data was collected using a questionnaire (see Appendix A) to be completed before the experiment, between runs and after completion of both runs. At the beginning of the study before the first run, participants were asked about their gender and age. In addition, the participants were asked to indicate their experience with VR via HMD and their experience of teleoperation with robots on a scale from 0 to 6.

4.6.1 Objective Data

Objective data included the measurement of the *Task Completion Time*, which measured how long it took the participant to complete a task. For this purpose, the time was started manually by the study supervisor as soon as the robot was in the correct position and the participant could see the virtual environment in the VR headset.
The time was also stopped by the study supervisor as soon as the participant signaled that they were approximately one meter in front of the checkerboard pattern.

The trajectory of the robot and the player in the virtual scene were also logged. For this purpose, the position and rotation of the object was saved in a file every second. The logging was implemented in the Unity application and the results were saved separately for each player for each navigation metaphor.

By exporting every last mesh of a model in OBJ format, movement sequences and position changes can later be combined with the trajectory data and visualized again at a later point in time.

4.6.2 Subjective Data

The subjective data included the measurements of several components to determine the usability and intuitiveness of the components. The usability of the overall system was assessed using the System Usability Scale (SUS) [51]. This is a standardized questionnaire to determine the usability of a system. An SUS questionnaire consists of 10 statements (see Table 4.4) relating to the user's experience of the system. The statements are rated on a 5-point Likert scale [52] from "Strongly disagree" to "Strongly agree".
The measurement of spatial presence in the experiment was conducted with MEC Spatial Presence Questionnaire (MEC-SPQ) [53]. The virtual environment is evaluated from the participant's perspective using the "Attention", "Spatial Situation" and "Presence" scales of the MEC-SPQ. Table 4.5 shows the statements used for MEC-SPQ.
The Single Ease Questionnaire (SEQ) [54] is used for the general evaluation of the subjective difficulty of the tasks in the user study. To measure the task workload, the NASA Task Load Index (NASA-TLX) [55] was used, which measures the participants' task workload in terms of mental demand, physical demand, temporal demand, performance, effort and frustration level. Participants assess the six dimensions of workload using a rating scale ranging from low to high, with scores generally ranging from 0 to 100. Participants then use pairwise comparisons to select which dimension is more important to the task workload being assessed, thus determining an importance for each item [56]. After completing both runs, users were asked to choose their preferred navigation metaphor and to explain their choice in a text field. The questionnaire ends with a free text form for general feedback on the system and the study.

I think that I would like to use this system frequently.
I found the system unnecessarily complex.
I thought the system was easy to use.
I think that I would need the support of a technical person to be able to use this system.
I found the various functions in this system were well integrated.
I thought there was too much inconsistency in this system.
I would imagine that most people would learn to use this system very quickly.
I found the system very cumbersome to use.
I felt very confident using the system.
I needed to learn a lot of things before I could get going with this system.

Table 4.4: Statements of the System Usability Scale (SUS). Participants rated the usability on a 5-point Likert scale (1=Strongly disagree, 5=Strongly agree). Statements from [51].

Attention
I devoted my whole attention to the robot.
I concentrated on the robot.
The robot captured my senses.
I dedicated myself completely to the robot.
Situation
I was able to imagine the arrangement of the spaces presented in the robot very well.
I had a precise idea of the spatial surroundings presented in the robot.
I was able to make a good estimate of the size of the presented space.
Even now, I still have a concrete mental image of the spatial environment.
Presence
I felt like I was actually there in the environment of the presentation.
It was as though my true location had shifted into the environment in the presentation.
I felt as though I was physically present in the environment of the presentation.
It seemed as though I actually took part in the action of the presentation.

Table 4.5: Statements of the MEC Spatial Presence Questionnaire. Participants rated the spatial presence on a 5-point Likert scale (1=Strongly disagree, 5=Strongly agree). Statements based on [53].

4.7 Results and Data Analysis

This chapter presents the results of the evaluation of the objective data and subjective data. The objective data is presented first, followed by the subjective data. For the evaluation, the distribution of the data was assessed using the Shapiro-Wilk test. If the data is normally distributed, the dependent t-test is used as a pair-wise test. For non-normally distributed data, the Wilcoxon test is used as a pair-wise test. A p-value of less than 0.05 is assessed as statistically significant.

4.7.1 Objective Results

Task Completion Time

The time was measured manually by starting and stopping within the Unity application. The two navigation metaphors *Direct Control* and *Point-and-Click* were compared in order to analyze their influence on efficiency. The results show that the task completion time is normally distributed ($p > 0.05$), which is the reason for using the dependent t-test. The results of the t-test showed a statistically significant difference between the two navigation metaphors ($p < 0.05$). With the *Point-and-Click* metaphor (150.00±25.88) the task was completed significantly faster than with the *Direct Control* metaphor (188.00±30.06). This indicates that navigation by setting waypoints is more efficient than direct control. Figure 4.4 shows the task completion times of both navigation metaphors.

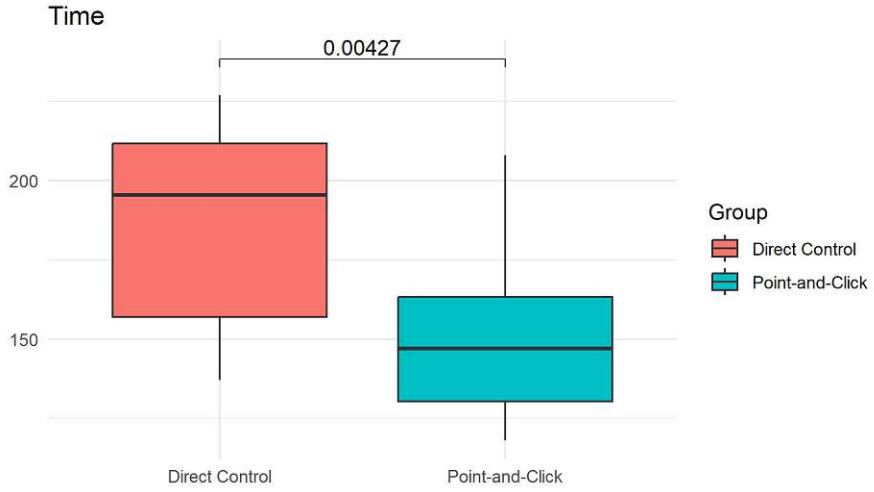


Figure 4.4: Overview of the *Task Completion Time* of both navigation metaphors in seconds.

Robot Trajectory

The trajectory of the robot was calculated using the stored position data of the robot. The normally distributed data ($p > 0.05$) showed a significant difference ($p < 0.05$) in the trajectory between the metaphors using the dependent t-test. The path was shorter with the *Point-and-Click* metaphor (23.51±0.92) than with the *Direct Control* metaphor (25.08±1.93). Figure 4.5 visualizes the comparison of the robot trajectory.

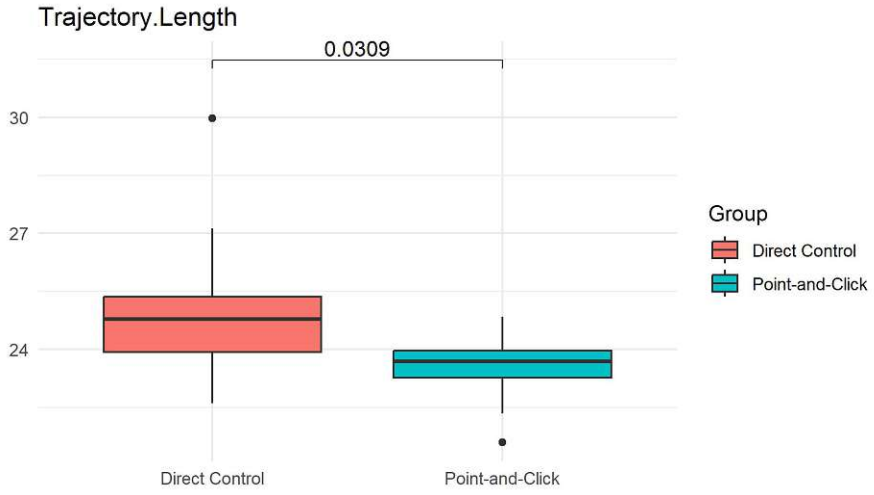


Figure 4.5: Overview of the robot trajectory results in meter.

Teleportation

It was recorded how often the users teleported in the scene with the respective metaphor. The Shapiro-Wilk test did not show a normal distribution of the data ($p < 0.05$). Therefore, the Wilcoxon test was used. No significant difference ($p > 0.05$) was found between the metaphors. This indicates that the choice of navigation metaphor has no influence on the frequency of teleportation. Figure 4.6 shows the average number of teleportations per metaphor.

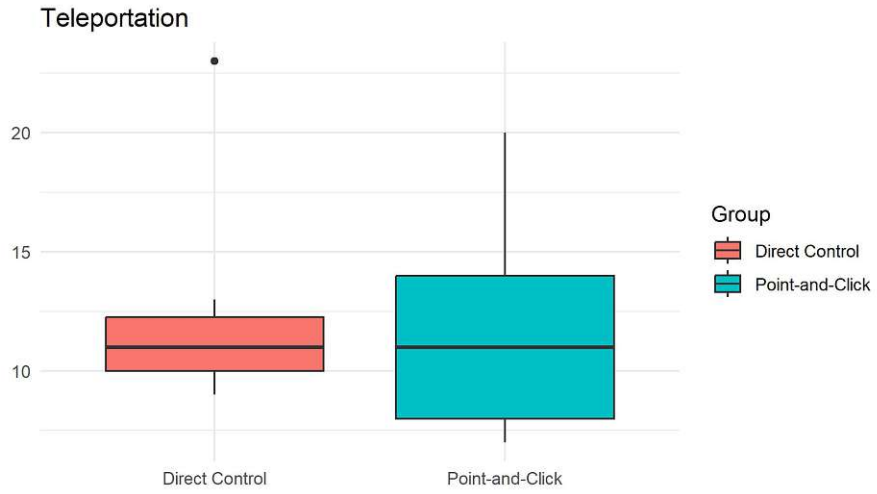


Figure 4.6: Overview shows the comparison of the number of user teleportation for both metaphors.

User-Robot Distance

The average distance between the robot and the user was calculated using the position data of both components stored every second in the virtual environment. The head position of the user and the robot position in the virtual space were taken into account. The non-normally distributed data ($p < 0.05$) of the distance between the robot and the user showed a statistically significant difference ($p < 0.05$). For example, users were closer to the robot in the *Point-and-Click* (1.73±0.33) metaphor compared to the *Direct Control* metaphor (2.14±0.65). This might be due to the fact that the *Direct Control* metaphor requires continuous control, with less time to adjust the user's position. In contrast, the *Point-and-Click* metaphor offers more opportunity to adjust the position, as the robot automatically moves to the waypoint after it has been set. Figure 4.7 shows the comparison of the user-robot distance for both navigation metaphors.

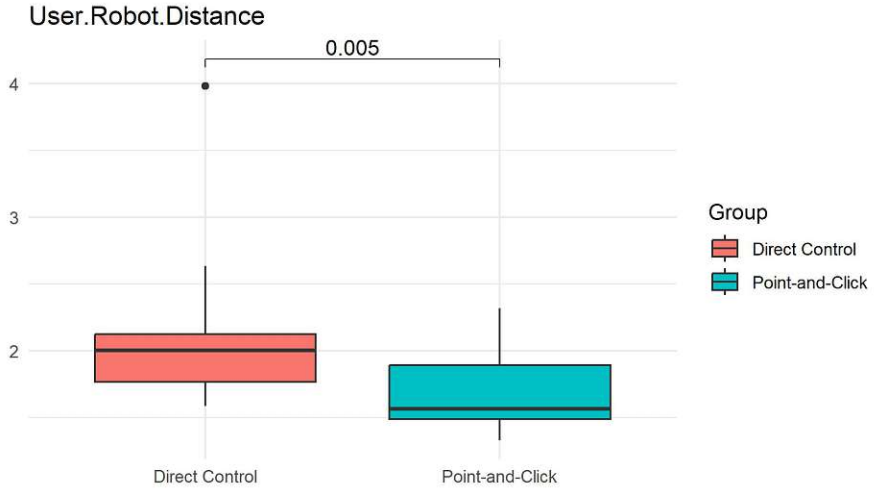


Figure 4.7: Overview of the *User-Robot Distance* for the metaphors.

4.7.2 Subjective Results

SUS

The results of the SUS questionnaire on the general usability of the system indicated a normal distribution of the data ($p > 0.05$). The dependent t-test revealed no statistically significant differences between the metaphors in terms of general usability. This indicates that both metaphors are perceived as similarly user-friendly. Figure 4.8 illustrates the SUS results of both metaphors.

Figure 4.8: Results of the System Usability Scale (SUS).

NASA-TLX

In the NASA-TLX workload analysis, all scales were evaluated. The navigation metaphors were compared using a pair-wise test. The overall scale showed normally distributed data ($p > 0.05$), whereas the remaining scales were not normally distributed. There was no statistically significant difference in any scale ($p > 0.05$). This indicates that the perceived workload in performing the task was similar for both metaphors. Figure 4.9 shows the comparison of the NASA-TLX results.

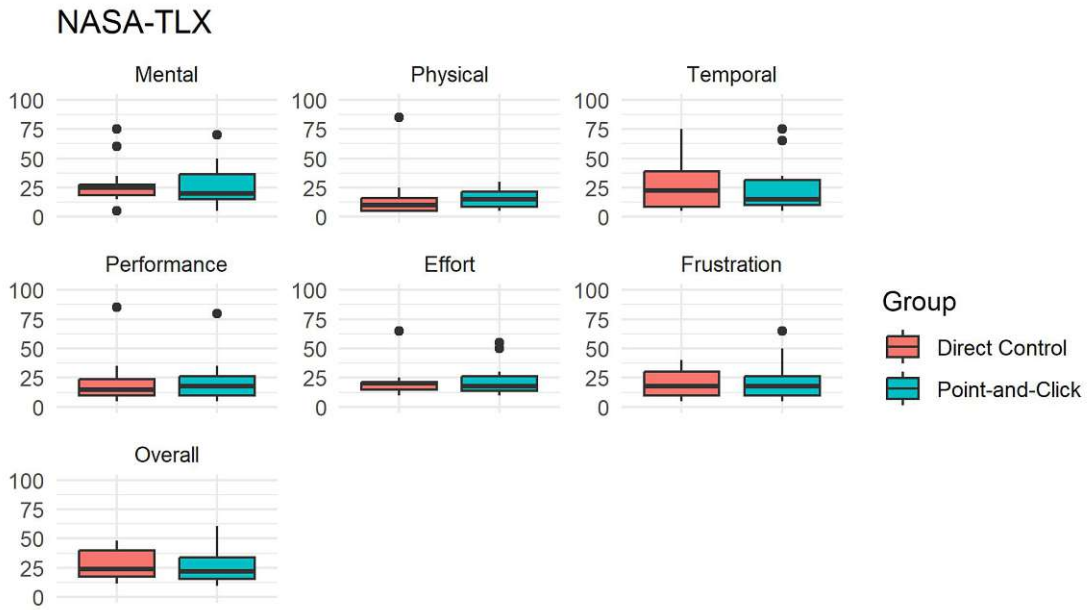


Figure 4.9: Results of NASA-TLX questionnaire comparing the workload of two navigation metaphors. Each subplot represents a different dimension of workload.

MEC-SPQ

The spatial presence questionnaire revealed normally distributed data ($p > 0.05$) for the attention, presence and situation factors. A statically significant difference ($p < 0.05$) was only found for the "situation" factor. Users seem to have paid more attention to the virtual world when using the *Direct Control* metaphor compared to the *Point-and-Click* metaphor. A comparison of the MEC-SPQ results can be found in Figure 4.10.

SEQ

The SEQ questionnaire for the general assessment of the subjective difficulty of completing the task showed no normally distributed data ($p < 0.05$). The results of both metaphors were exactly the same, meaning that no statistically significant difference ($p > 0.05$) could be determined. Figure 4.11 compares the SEQ results of the two navigation metaphors.

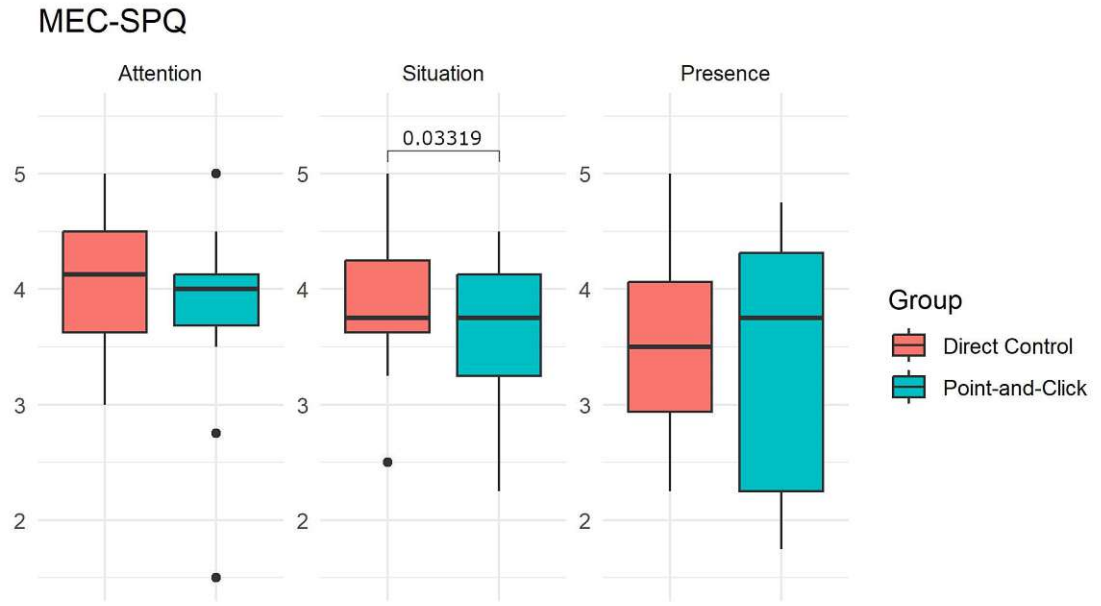


Figure 4.10: Results of the MEC Spatial Presence Questionnaire (MEC-SPQ) to measure spatial presence.

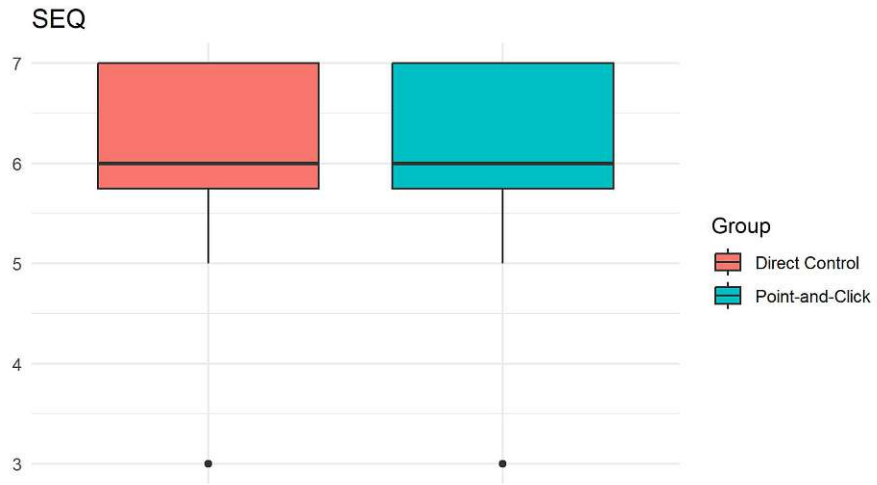


Figure 4.11: Results of Single Ease Questionnaire (SEQ).

4.8 Participant Feedback

After completing both runs, the participants were asked for final feedback at the end of the questionnaire. 7 of the 12 participants preferred the *Direct Control* metaphor, whereas the remaining 5 participants preferred the *Point-and-Click* metaphor. This means that 58% of the participants preferred the *Direct Control* metaphor, despite the

better efficiency of the *Point-and-Click* metaphor. From the reasons given in the free form fields, it can be concluded that a decisive reason was the continuous control over the robot. One participant mentioned the disadvantage that with the *Point-and-Click* method, the movement of the robot can no longer be aborted once the waypoint has been set. Other participants rated this aspect as an advantage, as they were better able to concentrate on their own navigation during the automatic movement of the robot. Some users rated the *Point-and-Click* metaphor as easier to use overall. With the *Direct Control* metaphor, the advantage of constant control over the robot was particularly emphasized. One participant also emphasized that the direct control felt more intuitive and made the system seem more realistic. However, one participant also found the continuous control of the *Direct Control* metaphor to be more cognitively demanding, as the robot has to be controlled manually on an ongoing basis.

CHAPTER 5

Discussion

This chapter discusses the results of the system developed and the user study conducted. First, the results of the user study are discussed on the basis of the objective and subjective data. The limitations are then explained.

5.1 Results

The user study compared two navigation metaphors for the teleoperation a mobile robot in VR. The metaphors were compared in terms of their efficiency and usability based on objective and subjective data. This showed both statically significant and non-significant differences, which are examined in more detail below.

The task completion time indicated a statistically significant difference in favor of the *Point-and-Click* metaphor, which enabled the user to complete the task quicker. It can be concluded that this control method enables more efficient navigation. One possible reason for this could be the lower interaction effort with this method. After setting the waypoint, the robot moves independently to the destination and the user can concentrate on their movement through teleportation or the robot's path planning. The efficiency is also reflected in the robot trajectory. A shorter path was found with the *Point-and-Click* metaphor, with a statically significant difference compared to the *Direct Control* metaphor.

Most subjective data, such as workload, general usability, spatial presence and subjective complexity, showed no statically significant differences between the metaphors. Only the "situation" factor in the MEC-SPQ questionnaire revealed a significant difference, with the *Direct Control* metaphor being rated more positively. In the *Direct Control* metaphor, users control the robot continuously, which can lead to greater attention to the virtual environment. As a result, users seem to feel more involved in the situation. Which indicates a higher situational awareness for this control metaphor. Some participants

also reported feeling more control over the robot with this metaphor and being able to perform precise movements. This is due to the fact that the user can control the movement at all times. In contrast, the *Point-and-Click* metaphor allows the user to place a waypoint to which the robot then moves independently. Once the waypoint has been set, the movement can no longer be stopped. Likewise, the position of the waypoint cannot be changed.

Although task completion time and robot trajectory were better with the *Point-and-Click* metaphor, no significant difference in workload was observed. Interestingly, despite poorer performance measured by the objective data, the users showed a higher perception of the spatial environment. This could indicate that because the *Direct Control* metaphor requires more control and therefore the user has to concentrate more on the environment.

Overall, the tasks were perceived as rather easy by the participants, regardless of the navigation metaphor used. Although there were no statically significant differences in the subjective data, both navigation metaphors showed good results overall in terms of usability and workload.

In the general feedback from the free text form of the questionnaire, the more direct control over the robot with the *Direct Control* metaphor was positively highlighted, but on the downside constant user input is of course needed. The *Point-and-Click* metaphor convinced by its ease of use. The points of criticism of the this metaphor were mainly related to the limited possibility of intervention during the autonomous movement of the robot to the next waypoint.

In summary, it can be concluded that both navigation metaphors can be considered functional and suitable for the teleoperation of a mobile robot in VR. While the *Point-and-Click* metaphor was objectively more efficient, the *Direct Control* metaphor was somewhat more convincing in terms of subjectively perceived control.

5.2 Limitations

The study revealed a number of limitations, which are explained in more detail in this section. The limitations include both the system itself and the implemented navigation metaphors. The identified limitations affect the user experience as well as the technical implementation and should be taken into account in future developments of the system.

One limitation refers to the *Point-and-Click* metaphor, where there is no possibility to interrupt the robot's movement or change the waypoint. Participants noted that there was no way to stop the movement when the waypoint was set.

The quality of the 3D reconstruction also showed problems. In some cases, the reconstructed mesh was not positioned correctly and there were visual artifacts and holes. This made orientation in virtual space difficult. These problems result, among other things, from the use of multiple models. However the implementation of multiple models was necessary to ensure the instantaneous display of meshes. Multiple models improved performance as the available system resources were used more efficiently.

The task in the user study was designed very straightforward, as the system reaches its limits when displaying more complex environments. The participants did not have to navigate around any obstacles or corners. The test environment was also very simple and only corresponded to realistic usage scenarios to a limited extent. Furthermore, the study took place in a controlled environment, which meant that it was less dynamic and complex.

In addition, the user study involved only 12 participants, which is a relatively small number and limits the significance of the results. A larger number of participants would be necessary to make more reliable statements about the user-friendliness and preference of the navigation metaphors.

CHAPTER 6

Summary

The thesis's results are summarized in this chapter, followed by ideas for potential future work in the field of robot teleoperation.

6.1 Conclusion

This thesis presented a system for the teleoperation of a mobile robot in a real environment using VR. Since the ongoing development in robotics leads to an increasing number of possible use cases, the human control of these robots is an exciting and promising field to explore. Providing the user with an immersive experience for navigating the robot efficiently via VR through a real environment was an essential challenge of the thesis.

This thesis focused on building a system for the Boston Dynamics Spot robot to be navigated remotely via VR through a real environment. The input data for the virtual representation of the robot's surroundings originates from a depth camera mounted on the Spot robot. To visualize the environment in VR, the Unity3D game engine was used. The user observes the robot's environment via an HTC VIVE Pro Eye headset and navigates it with HTC VIVE controllers. Teleportation via one of the controllers is used to position the user in the virtual surroundings.

With *Direct Control* metaphor and *Point-and-Click* metaphor, two different navigation methods were implemented in the system to navigate the robot. The two methods were evaluated in a subsequent user study.

The two navigation metaphors were compared in terms of their usability, intuitiveness, and efficiency. In the study, the users were assigned to navigate the robot from a starting point through a hallway to a target point. In two test runs, the users had to use both navigation metaphors to complete the task. Objective and subjective feedback data were collected via several functions in Unity and a questionnaire, respectively.

The objective results of the user study show that the task was completed significantly faster with the *Point-and-Click* metaphor. Additionally, the user moved the robot on a shorter path to the target point with this metaphor. The proximity between the user and the robot was shorter using the *Point-and-Click* metaphor.

The evaluation of the subjective data showed only one statistically significant difference between the two metaphors, namely that *Direct Control* indicates a better perception of the situation. The remaining results for perceived ease of use, workload, and difficulty showed no significant differences.

It can be concluded that both navigation metaphors represent user-friendly and effective control approaches for the teleoperation of a mobile robot in VR.

6.2 Future Work

Besides the positive results, the work presented also shows opportunities for improvement. One key aspect would be increasing the number of participants in a future user study. In order to be able to make better statements about the usability of the system, future studies in this field of research should be conducted with a larger and more diverse user group.

The complexity of the task could be increased in order to better represent practical applications. This includes the integration of obstacles as well as the execution of tasks over longer distances. Tasks of longer duration in future studies could provide insights into the effects of fatigue.

Some weaknesses were identified in the area of real-time 3D reconstruction. Using an improved visualization of 3D meshes in future studies could exploit more of the system's potential. An improvement should focus in particular on the positional accuracy of the meshes. A detailed representation of the environment also facilitates orientation and confidence in the system.

Further potential for improvement concerns the flexibility of the *Point-and-Click* metaphor. In the user study, it was not possible to cancel a running command or change the waypoint. A feature to interrupt or adjust the movement would potentially increase the user's control over the robot.

This work has created a solid basis for future developments of teleoperating mobile robots with immersive interfaces. With the above suggested improvements in future work, the understanding and use of teleoperation can be taken to the next level.

Overview of Generative AI Tools Used

List of Figures

List of Tables

Bibliography

[1] M. A. Goodrich and A. C. Schultz, "Human-Robot Interaction: A Survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[2] A. Batistute, E. Santos, K. Takieddine, P. M. Lazari, L. Giane Da Rocha, and K. C. Teixeira Vivaldini, "Extended Reality for Teleoperated Mobile Robots," in *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, (Natal, Brazil), pp. 19–24, IEEE, Oct. 2021.

[3] K. A. Szczurek, R. M. Prades, E. Matheson, J. Rodriguez-Nogueira, and M. D. Castro, "Multimodal Multi-User Mixed Reality Human–Robot Interface for Remote Operations in Hazardous Environments," *IEEE Access*, vol. 11, pp. 17305–17333, 2023.

[4] J. T. Isaacs, K. Knoedler, A. Herdering, M. Beylik, and H. Quintero, "Teleoperation for Urban Search and Rescue Applications," *Field Robotics*, vol. 2, pp. 1177–1190, June 2022.

[5] R. Murphy, "Human–Robot Interaction in Rescue Robotics," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 34, pp. 138–153, May 2004.

[6] W. J. Meijer, A. C. Kemmeren, J. M. van Bruggen, T. Haije, J. E. Fransman, and J. D. van Mil, "Situational Graphs for Robotic First Responders: an application to dismantling drug labs," Apr. 2024. arXiv:2404.17395 [cs].

[7] H. Stedman, B. B. Kocer, N. Van Zalk, M. Kovac, and V. M. Pawar, "Evaluating Immersive Teleoperation Interfaces: Coordinating Robot Radiation Monitoring Tasks in Nuclear Facilities," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, (London, United Kingdom), pp. 11972–11978, IEEE, May 2023.

[8] M. Wonsick, T. Kelestemur, S. Alt, and T. Padir, "Telemanipulation via Virtual Reality Interfaces with Enhanced Environment Models," in *2021 IEEE/RSJ In-*

ternational Conference on Intelligent Robots and Systems (IROS), (Prague, Czech Republic), pp. 2999–3004, IEEE, Sept. 2021.

[9] S. Livatino, D. C. Guastella, G. Muscato, V. Rinaldi, L. Cantelli, C. D. Melita, A. Caniglia, R. Mazza, and G. Padula, "Intuitive Robot Teleoperation Through Multi-Sensor Informed Mixed Reality Visual Aids," *IEEE Access*, vol. 9, pp. 25795–25808, 2021.

[10] V. Villani, B. Capelli, and L. Sabattini, "Use of Virtual Reality for the Evaluation of Human-Robot Interaction Systems in Complex Scenarios," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, (Nanjing), pp. 422–427, IEEE, Aug. 2018.

[11] P. Stotko, S. Krumpen, M. Schwarz, C. Lenz, S. Behnke, R. Klein, and M. Weinmann, "A VR System for Immersive Teleoperation and Live Exploration with a Mobile Robot," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Macau, China), pp. 3630–3637, IEEE, Nov. 2019.

[12] J. E. Solanes, A. Muñoz, L. Gracia, and J. Tornero, "Virtual Reality-Based Interface for Advanced Assisted Mobile Robot Teleoperation," *Applied Sciences*, vol. 12, p. 6071, June 2022.

[13] S. Holder and L. Stirling, "Effect of Gesture Interface Mapping on Controlling a Multi-degree-of-freedom Robotic Arm in a Complex Environment," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 64, pp. 183–187, Dec. 2020.

[14] E. Solly and A. Aldabbagh, "Gesture Controlled Mobile Robot," in *2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, (Istanbul, Turkiye), pp. 1–6, IEEE, June 2023.

[15] S. Ahmad, M. Alhammadi, A. Alamoodi, A. Alnuaimi, S. Alawadhi, and A. Alsumaiti, "On the Design and Fabrication of a Voice-controlled Mobile Robot Platform:," in *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics*, pp. 101–106, SCITEPRESS - Science and Technology Publications, 2021.

[16] A. Poncela and L. Gallardo-Estrella, "Command-based voice teleoperation of a mobile robot via a human-robot interface," *Robotica*, vol. 33, pp. 1–18, Jan. 2015.

[17] D. W. Hainsworth, "Teleoperation User Interfaces for Mining Robotics," *Autonomous robots*, vol. 11, no. 1, pp. 19–28, 2001.

[18] G. Baker, T. Bridgwater, P. Bremner, and M. Giuliani, "Towards an immersive user interface for waypoint navigation of a mobile robot," Mar. 2020. arXiv:2003.12772 [cs].

[19] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, and S. Tellex, "Comparing robot grasping teleoperation across desktop and virtual reality with ros reality," in *Robotics Research: The 18th International Symposium ISRR*, pp. 335–350, Springer, 2019.

[20] M. R. Endsley, "Situation Awareness in Aircraft Systems: Symposium Abstract," *Proceedings of the Human Factors Society Annual Meeting*, vol. 32, pp. 96–96, Oct. 1988.

[21] R. Hetrick, N. Amerson, B. Kim, E. Rosen, E. J. D. Visser, and E. Phillips, "Comparing Virtual Reality Interfaces for the Teleoperation of Robots," in *2020 Systems and Information Engineering Design Symposium (SIEDS)*, (Charlottesville, VA, USA), pp. 1–7, IEEE, Apr. 2020.

[22] M. Wonsick and T. Padir, "A Systematic Review of Virtual Reality Interfaces for Controlling and Interacting with Robots," *Applied Sciences*, vol. 10, p. 9051, Dec. 2020.

[23] J. C. Garcia, B. Patrao, L. Almeida, J. Perez, P. Menezes, J. Dias, and P. J. Sanz, "A Natural Interface for Remote Operation of Underwater Robots," *IEEE Computer Graphics and Applications*, vol. 37, pp. 34–43, Jan. 2017.

[24] J. D. Moss and E. R. Muth, "Characteristics of Head-Mounted Displays and Their Effects on Simulator Sickness," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 53, pp. 308–319, June 2011.

[25] M. E. Walker, M. Gramopadhye, B. Ikeda, J. Burns, and D. Szafir, "The Cyber-Physical Control Room: A Mixed Reality Interface for Mobile Robot Teleoperation and Human-Robot Teaming," in *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, (Boulder CO USA), pp. 762–771, ACM, Mar. 2024.

[26] M. E. Walker, H. Hedayati, and D. Szafir, "Robot Teleoperation with Augmented Reality Virtual Surrogates," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, (Daegu, Korea (South)), pp. 202–210, IEEE, Mar. 2019.

[27] W. Si, T. Zhong, N. Wang, and C. Yang, "A multimodal teleoperation interface for human-robot collaboration," in *2023 IEEE International Conference on Mechatronics (ICM)*, (Loughborough, United Kingdom), pp. 1–6, IEEE, Mar. 2023.

[28] S. Bonaiuto, A. Cannavo, G. Piumatti, G. Paravati, and F. Lamberti, "Tele-operation of Robot Teams: A Comparison of Gamepad-, Mobile Device and Hand Tracking-Based User Interfaces," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, (Turin), pp. 555–560, IEEE, July 2017.

[29] V. Pavlovic, R. Sharma, and T. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 677–695, July 1997.

[30] J. Paterson and A. Aldabbagh, "Gesture-Controlled Robotic Arm Utilizing OpenCV," in *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, (Ankara, Turkey), pp. 1–6, IEEE, June 2021.

[31] A. Martín-Barrio, J. J. Roldán, S. Terrile, J. Del Cerro, and A. Barrientos, "Application of immersive technologies and natural language to hyper-redundant robot teleoperation," *Virtual Reality*, vol. 24, pp. 541–555, Sept. 2020.

[32] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[33] S. N. Young and J. M. Peschel, "Review of Human–Machine Interfaces for Small Unmanned Systems With Robotic Manipulators," *IEEE Transactions on Human-Machine Systems*, vol. 50, pp. 131–143, Apr. 2020.

[34] T. Piumsomboon, A. Day, B. Ens, Y. Lee, G. Lee, and M. Billinghurst, "Exploring enhancements for remote mixed reality collaboration," in *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, (Bangkok Thailand), pp. 1–5, ACM, Nov. 2017.

[35] M. Walker, T. Phung, T. Chakraborti, T. Williams, and D. Szafir, "Virtual, Augmented, and Mixed Reality for Human-robot Interaction: A Survey and Virtual Design Element Taxonomy," *ACM Transactions on Human-Robot Interaction*, vol. 12, pp. 1–39, Dec. 2023.

[36] J. I. Lipton, A. J. Fay, and D. Rus, "Baxter's Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing," *IEEE Robotics and Automation Letters*, vol. 3, pp. 179–186, Jan. 2018.

[37] H. Bavle, J. L. Sanchez-Lopez, C. Cimarelli, A. Tourani, and H. Voos, "From SLAM to Situational Awareness: Challenges and Survey," *Sensors*, vol. 23, p. 4849, May 2023.

[38] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*, vol. 200. Springer, 2008.

[39] "Spot core payload (legacy)." Accessed: 2025-04-01. Available: `https://support.bostondynamics.com/s/article/Spot-Core-Payload-Legacy-72064`.

[40] "'image: Flaticon.com': This illustration has been designed using resources from flaticon.com."

[41] M. Servi, A. Profili, R. Furferi, and Y. Volpe, "Comparative Evaluation of Intel RealSense D415, D435i, D455, and Microsoft Azure Kinect DK Sensors for 3D Vision Applications," *IEEE Access*, vol. 12, pp. 111311–111321, 2024.

[42] "Intel® RealSense™ Depth Camera D435i." Accessed: 2025-02-23. Available: `https://www.intelrealsense.com/depth-camera-d435i/`.

[43] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," Jan. 2018.

[44] "Build from source." Accessed: 2025-04-02. Available: https://www.open3d.org/docs/release/compilation.html.

[45] "Dense RGB-D SLAM." Accessed: 2025-03-10. Available: https://www.open3d.org/docs/latest/tutorial/t_reconstruction_system/dense_slam.html.

[46] W. Dong, Y. Lao, M. Kaess, and V. Koltun, "ASH: A modern framework for parallel spatial hashing in 3D perception," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2022.

[47] "Plane segmentation." Accessed: 2025-04-02. Available: https://www.open3d.org/docs/latest/tutorial/Basic/pointcloud.html#Plane-segmentation.

[48] V. Mansur, S. Reddy, S. R, and R. Sujatha, "Deploying Complementary filter to avert gimbal lock in drones using Quaternion angles," in *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, (Greater Noida, India), pp. 751–756, IEEE, Oct. 2020.

[49] A. Shahbaz Badr and R. De Amicis, "An empirical evaluation of enhanced teleportation for navigating large urban immersive virtual environments," *Frontiers in Virtual Reality*, vol. 3, p. 1075811, Jan. 2023.

[50] "Raumübersicht." Accessed: 2025-04-19. Available: https://www.tuwien.at/fileadmin/Assets/dienstleister/gebaeude_und_technik/FS/Plaene_2/Favoritenstrasse_9-11_1040_HA-HI_IP_09012020.pdf.

[51] J. Brooke, "SUS - A quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[52] R. Likert, "A technique for the measurement of attitudes.," *Archives of psychology*, 1932.

[53] P. Vorderer, W. Wirth, F. R. Gouveia, F. Biocca, T. Saari, Futz Jäncke, S. Böcking, H. Schramm, A. Gysbers, T. Hartmann, *et al.*, "MEC spatial presence questionnaire (MEC-SPQ): Short documentation and instructions for application," *Report to European Community, Project Presence MEC (IST-2001-37661)*, 2004.

[54] "Measuring task usability: The Single Ease Question (SEQ)." [Online]. Accessed: 2025-03-14. Available: https://trymata.com/blog/measuring-task-usability-the-single-ease-question/.

[55] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research," in *Advances in Psychology*, vol. 52, pp. 139–183, Elsevier, 1988.

[56] M. Schrum, M. Ghuy, E. Hedlund-botti, M. Natarajan, M. Johnson, and M. Gombolay, "Concerning Trends in Likert Scale Usage in Human-robot Interaction: Towards Improving Best Practices," *ACM Transactions on Human-Robot Interaction*, vol. 12, pp. 1–32, Sept. 2023.

Appendix

Appendix A

Spot 3D reconstruction and navigation

* Erforderlich

Demographics

1. UserID *

Der Wert muss eine Zahl sein.

2. What is your gender? *

- ◯ Woman
- ◯ Man
- ◯ Non-binary
- ◯ Prefer not to disclose
- ◯ Prefer to self describe

3. Please self describe your gender

4. Age *

Geben Sie eine Zahl größer als 17 ein.

5. Please answer the following questions *

	0 Never	1	2	3	4	5	6 Regularly
Have you experienced virtual reality with a head mounted display?	◯	◯	◯	◯	◯	◯	◯
Have you experienced teleoperation with robots?	◯	◯	◯	◯	◯	◯	◯

Condition 1

Please answer the following questions

6. Condition tested (filled by experimenter) *

- ◯ Steering
- ◯ Pointing

7. On a scale of 20, ranging from 0 (no sickness at all) to 20 (frank sickness) how are you feeling right now? (focus on nausea, general discomfort, and stomach problems) *

Die Zahl muss zwischen 0 und 20 liegen

8. Please answer the following statement

	Very Difficult	Difficult	Somewhat Difficult	Neither Difficult or Easy	Somewhat Easy	Easy	Very Easy
Overall, how difficult or easy did you find this task?	◯	◯	◯	◯	◯	◯	◯

9. Please answer the following statements *

	Strongly disagree	disagree	Neither agree or disagree	Agree	Strongly Agree
I devoted my whole attention to the robot	◯	◯	◯	◯	◯
I concentrated on the robot	◯	◯	◯	◯	◯
The robot captured my senses.	◯	◯	◯	◯	◯
I dedicated myself completely to the robot	◯	◯	◯	◯	◯
I was able to imagine the arrangement of the spaces presented in the robot very well.	◯	◯	◯	◯	◯
I had a precise idea of the spatial surroundings presented in the robot	◯	◯	◯	◯	◯
I was able to make a good estimate of the size of the presented space.	◯	◯	◯	◯	◯
Even now, I still have a concrete mental image of the spatial environment.	◯	◯	◯	◯	◯
I felt like I was actually there in the environment of the presentation.	◯	◯	◯	◯	◯
It was as though my true location had shifted into the environment in the presentation.	◯	◯	◯	◯	◯
I felt as though I was physically present in the environment of the presentation.	◯	◯	◯	◯	◯
It seemed as though I actually took part in the action of the presentation.	◯	◯	◯	◯	◯

10. Please answer the following statements *

	Strongly disagree	disagree	Neither agree or disagree	Agree	Strongly Agree
I think that I would like to use this system frequently.	◯	◯	◯	◯	◯
I found the system unnecessarily complex.	◯	◯	◯	◯	◯
I thought the system was easy to use.	◯	◯	◯	◯	◯
I think that I would need the support of a technical person to be able to use this system.	◯	◯	◯	◯	◯
I found the various functions in this system were well integrated.	◯	◯	◯	◯	◯
I thought there was too much inconsistency in this system.	◯	◯	◯	◯	◯
I would imagine that most people would learn to use this system very quickly.	◯	◯	◯	◯	◯
I found the system very cumbersome to use.	◯	◯	◯	◯	◯
I felt very confident using the system.	◯	◯	◯	◯	◯
I needed to learn a lot of things before I could get going with this system.	◯	◯	◯	◯	◯

11. NASA TLX (https://www.keithv.com/software/nasatlx/nasatlx.html) Paste the results of the questionnaire in the field textbox *

Condition 2

Please answer the following questions

12. Condition tested (filled by experimenter) *

- ◯ Steering
- ◯ Pointing

13. On a scale of 20, ranging from 0 (no sickness at all) to 20 (frank sickness) how are you feeling right now? (focus on nausea, general discomfort, and stomach problems) *

Die Zahl muss zwischen 0 und 20 liegen

14. Please answer the following statement

	Very Difficult	Difficult	Somewhat Difficult	Neither Difficult or Easy	Somewhat Easy	Easy	Very Easy
Overall, how difficult or easy did you find this task?	◯	◯	◯	◯	◯	◯	◯

15. Please answer the following statements *

	Strongly disagree	disagree	Neither agree or disagree	Agree	Strongly Agree
I devoted my whole attention to the robot	◯	◯	◯	◯	◯
I concentrated on the robot	◯	◯	◯	◯	◯
The robot captured my senses.	◯	◯	◯	◯	◯
I dedicated myself completely to the robot	◯	◯	◯	◯	◯
I was able to imagine the arrangement of the spaces presented in the robot very well.	◯	◯	◯	◯	◯
I had a precise idea of the spatial surroundings presented in the robot	◯	◯	◯	◯	◯
I was able to make a good estimate of the size of the presented space.	◯	◯	◯	◯	◯
Even now, I still have a concrete mental image of the spatial environment.	◯	◯	◯	◯	◯
I felt like I was actually there in the environment of the presentation.	◯	◯	◯	◯	◯
It was as though my true location had shifted into the environment in the presentation.	◯	◯	◯	◯	◯
I felt as though I was physically present in the environment of the presentation.	◯	◯	◯	◯	◯
It seemed as though I actually took part in the action of the presentation.	◯	◯	◯	◯	◯

16. Please answer the following statements *

	Strongly disagree	disagree	Neither agree or disagree	Agree	Strongly Agree
I think that I would like to use this system frequently.	◯	◯	◯	◯	◯
I found the system unnecessarily complex.	◯	◯	◯	◯	◯
I thought the system was easy to use.	◯	◯	◯	◯	◯
I think that I would need the support of a technical person to be able to use this system.	◯	◯	◯	◯	◯
I found the various functions in this system were well integrated.	◯	◯	◯	◯	◯
I thought there was too much inconsistency in this system.	◯	◯	◯	◯	◯
I would imagine that most people would learn to use this system very quickly.	◯	◯	◯	◯	◯
I found the system very cumbersome to use.	◯	◯	◯	◯	◯
I felt very confident using the system.	◯	◯	◯	◯	◯
I needed to learn a lot of things before I could get going with this system.	◯	◯	◯	◯	◯

17. NASA TLX (https://www.keithv.com/software/nasatlx/nasatlx.html) Paste the results of the questionnaire in the field textbox *

Post Experiment

Please answer the following questions

18. Which condition did you prefer? *

- ◯ Steering
- ◯ Pointing

19. Why did you prefer this interface?

20. General comments

Credits

Icons used in Figure 3.1, Figure 3.2 and Figure 3.6 downloaded from Flaticon.com