FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Juho Kalliokoski

HI-DWA: HUMAN INFLUENCED DYNAMIC WINDOW APPROACH FOR SHARED CONTROL OF A TELEPRESENCE ROBOT

Master's Thesis
Degree Programme in Computer Science and Engineering
October 2022

Kalliokoski J. (2022) HI-DWA: Human Influenced Dynamic Window Approach for Shared Control of a Telepresence Robot. University of Oulu, Degree Programme in Computer Science and Engineering, 38 p.

ABSTRACT

This thesis introduces a shared control mechanism, which allows the user to modify the path of a telepresence robot. The robot is capable of autonomously navigating to a goal predefined by the user, but the user might still want to adjust the path, for example, to go further away from other people or robots, steer away from a wall or go closer to interesting landmarks they want to see on the way. The thesis proposes Human-Influenced Dynamic Window Approach (HI-DWA), a shared control method aimed for immersive telepresence robots based on Dynamic Window Approach (DWA). The proposed method was compared with switching between autonomous navigation and manual control in a user study (N=32), where participants controlled a simulated telepresence robot in Virtual Reality (VR) using various control methods. Results showed that the users reached their goal faster using HI-DWA controller and found it less demanding and easier to use. However, preference between the two methods was split equally. Qualitative analysis revealed that a major reason for the participants that preferred switching between the two modes was the feeling of control. The study also analyzed the effect of different input methods using the HI-DWA, joystick and gesture, on the preference and perceived workload. Even though the gesture based HI-DWA was found significantly harder and more demanding to use, it was preferred more than the proposed method using joystick, or the switching.

Keywords: immersive telepresence, autonomous navigation, shared control, virtual reality

Kalliokoski J. (2022) Käyttäjävaikutteinen DWA-pohjainen etäläsnäolorobotin jaettu ohjaus. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 38 s.

TIIVISTELMÄ

Tämä diplomityö esittelee jaetun ohjauksen mekanismin, joka mahdollistaa käyttäjän muuttamaan etäläsnäolorobotin kulkemaa reittiä. Robotti kykenee navigoimaan käyttäjän määrittelemään määränpäähän, mutta käyttäjä saattaa haluta vaikuttaa sen kulkemaan reittiin. Esimerkiksi käyttäjä saatta haluta mennä kauemmaksi muista ihmisistä tai roboteista, kääntää kauemmaksi seinästä tai mennä lähemmäksi mielenkiintoisia kiintopisteitä, joita he haluavat nähdä reitin varrella. Diplomityö esittelee jaettuun ohjaukseen tarkoitetun HI-DWA-menetelmän, , joka on suunniteltu immersiivisille etäläsnäolo roboteille ja joka pohjautuu DWA-menetelmään. Ehdotettua menetelmää verrataan menetelmää, jossa käyttäjä voi vaihdella autonomisen ja manuaalisen ohjauksen välillä. Vertailu tehtiin käyttäjäkokeessa (N=32), jossa käyttäjät ohjasivat simuloitua etäläsnäolorobottia virtuaalitodellisuudessa erilaisia ohjausmenetelmiä käyttäen. Tutkimuksen tulokset näyttivät, että käyttäjät pääsivät tavoitepaikkaansa nopeammin käyttäen HI-DWA-metodia ja kokivat sen helpommaksi käyttää ja vähemmän vaativaksi. Kuitenkin, kysymykseen "kumpaa menetelmää suosit"vastaukset jakaantuivat tasan. Kvalitatiivinen analyysi osoitti merkittävimmäksi syyksi autonomisen ja manuaalisen ohjauksen välillä vaihtelun suosioon käyttäjien kokema täysi hallinta robotin liikkeisiin. Tutkimus tutki myös erilaisten ohjausmenetelmien, ohjaussauvan sekä eleohjauksen, vaikutusta HI-DWA-menetelmää käyttäessä suosioon sekä koettuun työtaakkaan. Vaikka eleohjaus koettiin merkittävän verran hankalammaksi sekä vaativammaksi käyttää, melkein puolet käyttäjistä suosivat sitä verrattaessa ohjaussauvaan tai autonomian tason vaihteluun.

Avainsanat: immersiivinen etäläsnäolo, autonominen navigointi, jaettu ohjaus, virtuaali todellisuus

TABLE OF CONTENTS

FOREWORD

I would first like to thank my main supervisor Başak Sakçak, along with the other supervisors, Kalle Timperi and Markku Suomalainen. With their continuous support with the development and writing progress I feel like I have learned a lot about conducting research and writing proper scientific text. I would like to thank Alexis Chambers for her efforts in conducting the study and collecting the participants. I am also really thankful to my friends and family for giving me motivation and support on my journey through this whole progress. Last but not the least I would like to thank the Center for Ubiquitous Computing for letting me be part of their research group and giving me the opportunity for conducting this thesis.

Oulu, October 31st, 2022

Juho Kalliokoski

LIST OF ABBREVIATIONS

SJ	Shared Control-Joystick
SW	Switching Control
SG	Shared Control-Gesture
VR	Virtual Reality
HMD	Head-Mounted Display
DWA	Dynamic Window Approach
ROS2	Robot Operating System 2
HI-DWA	Human-Influenced Dynamic Window Approach

1. INTRODUCTION

Telepresence robots have become increasingly relevant due to the need of being able to work from home. Being able to control a robot in another location, to see the environment around the robot, and to interact with it gives possibilities to meet and interact with people at a physical location even when it is not possible for everyone to get to the meeting location by themselves. They can be used in important personal milestones such as birthdays, weddings, or graduations, but also in business meetings and factory tours. At the moment the commercial telepresence robots, such as Double [1] and GoBe [2], are essentially a Skype-on-wheels, which is sufficient for simple video conferencing. However, a recent study showed that in a hybrid meeting, people who were present only virtually did not participate in conversations as much, and found it harder to partake in a shared task compared to people who were physically present [3]. These limitations increased the interest in immersive robotic telepresence for which the user embodies a mobile robot in a remote location through a Head-Mounted Display (HMD). Being able to look around and get immersed in the experience provided by such a telepresence robot has the potential to overcome the gap between physically and virtually present people. However, VR has its own issues such as complexity and proneness to issues such as VR sickness.

Even the “regular" telepresence robots have their own limitations and problems with autonomous navigation, such as the problem of human-aware navigation [4]. In addition to these, immersive telepresence robots have their own challenges that need to be addressed. In addition to safely avoiding people in the environment, a telepresence robot should also ensure that the person on board feels comfortable with the robot’s motion and the path taken by it. This is true especially with immersive telepresence robots, as it has been shown that the user comfort is one of the key factors for the immersion when using VR [5]. Since aspects related to comfort and preference vary from one person to another, it seems natural to endow the user with effortless methods to make slight changes to the robot’s autonomous path. These allow addressing issues such as going too close to people or too far from points of interest not known to the autonomous planner.

Shared control refers to systems for which the human and the robot are cooperating to achieve a task. Telepresence robots can benefit from the shared control system, since controlling a robot from a remote location typically involves problems such as delays in the communication and video feed, and difficulties in recognizing obstacles around the robot due to limited visibility. Shared control provide the ease of the autonomous control without losing total control over the robot’s motions and giving a possibility to do slight changes to the path according to the user’s own liking.

A key factor contributing to VR sickness is the robot motion, which underlines the importance of the choice of the control method when using immersive telepresence. While the other VR experiences can use teleporting and snapping rotations to reduce sickness, it is not applicable when controlling a telepresence robot. Giving the user complete manual control over the robot through joystick commands is the simplest method to enable the user to affect the robot motion. However, there is evidence suggesting that people find manual controls tiring [6], which motivates further research towards autonomous or semi-autonomous methods. An example of these is the “waypoint navigation" found in the commercial telepresence robot Double for which

the user points at a point within the visible area, and the robot navigates towards there autonomously. This type of navigation is also shown to be useful for immersive telepresence [7].

Even though shared control is a well-known problem in robotics, there is limited research on aspects related to telepresence, either for “regular" or immersive telepresence. This study focuses on the problem of taking the user preference into account for the autonomous navigation of a telepresence robot to account for the variability in the user preferences. It proposes a shared control method for an immersive telepresence robot, that allows the user to influence the trajectory executed by the robot with minimal effort. The proposed system, HI-DWA, is an adaptation of the popular DWA [8], which searches for the best control input that optimizes a relevant objective by integrating the system dynamics for each admissible control input and scoring the resulting trajectories. The key idea of HI-DWA is to penalize the deviation from the user input. Thus, we retain the collision avoidance property of DWA, while allowing the user to influence the robot's motion. To evaluate the proposed method, a study (N=32) was conducted, in which the participants wearing HMDs compared the proposed control method with switching between manual and autonomous modes (a method often used in commercial telepresence robots, such as Double 3) for a simulated robot in a virtual environment that mimics immersive telepresence. They were able to give inputs to the robot using a joystick included in Oculus Quest 2 controller. They were encouraged to alter the path of the robot. The results showed that even though participants found the proposed method easier, there was no difference in preference. Further analyses to interpret this result indicated that a major reason for seeing no difference in preference is that many users liked having full control over the robot's motion. Also, for the use case of immersive telepresence, we found that adjusting the path using gestures (moving hand) was preferred over joystick.

In this chapter we have introduced the research problem and the motivation of this study. Chapter 2 presents literature background of robotic telepresence, VR and VR sickness, shared control systems, planning methods and robot controllers used in autonomous navigation. We also motivate the present study in light of existing literature. Chapter 3 introduces the proposed control method on a technical level. Chapter 4 presents the study specific implementation and the tools used for the implementation. It also presents the three conditions used in the study, the physical and virtual environments used, as well as the hypotheses, participants and the measures of the study, and the study procedure itself. Chapter 5 presents the results of the study together with the discussion. Finally, Chapter 6 concludes the thesis.

2. RELATED WORK

In this chapter we introduce in more depth the underlying concepts relevant to our study. In addition to telepresence and different control methods, these include also VR, planning and controllers. We also motivate the present study in light of this literature.

2.1. Robotic Telepresence

In robotic telepresence, a robot in the local environment is connected to and controlled by an operator in a remote location [9]. The robot is equipped with a video conferencing system, which the operator can use to see the robot's environment and move the robot around. The term *"telepresence"* was introduced by Marvin Minsky in 1980 [10]. He described it as the possibility to use sensors connected to one's body to control a robot in a remote location. Originally Minsky imagined telepresence being used to control machines at a factory from a remote location. Lately both researchers and commercial implementations have concentrated heavily on the social aspects of telepresence. This takes the form of a mobile robot with a telecommunication hardware and a controller attached to it. These are used, for example, by students who otherwise could not attend lectures [11], social interaction [9], and telemonitoring on the basis of medical consultation [12]. These kinds of telepresence robots usually consists of a mobile platform with a camera connected to it. The operator connects to, and controls the robot from a remote location and views the robots surroundings through a flat screen. Notably, a recent study showed that when attending in a hybrid meeting using this kind of telepresence robot, participants did not communicate as much as physically present participants and they had difficulties with group tasks [3]. These problems could be solved using *immersive telepresence*, a type of telepresence where the normal camera and the flat monitor are replaced with 360 degree camera and a HMD. Using this kind of setup can increase the immersion, and make the user to feel more like they are *there* [13]. However, immersive telepresence has its own setbacks such as VR sickness [14, 15, 16]. Studies have shown that being moved around while in VR can cause VR sickness [17]. This has prompted VR application developers to avoid continuous movement in VR, for example by replacing movement with teleportation [18]. This, however, is not possible in immersive telepresence, as the physical body of the robot does not allow moving the viewpoint around long distances in an instant. With the immersive telepresence robot moving continuously around, there is a need to decrease sickness by making other parts of the movement as comfortable as possible. For example, it has been shown that the distance to the walls, corners and other obstacles plays a big role in user experience, when being moved around by an immersive telepresence robot [19]. This kind of user experience is not often taken into consideration when developing controllers for traditional telepresence robots.

2.2. Virtual Reality

VR has been studied for a long time already. Sutherland presented an early version of HMD already in 1968 [20]. Only few new applications emerged, such as Nintendo Virtual Boy in 1995 [21], the technology did not take off until the Oculus Rift was introduced in 2012 bringing the so-called second wave of VR [22]. The advances in technology made it possible to create cheaper, lighter, and more powerful VR headsets and powerful enough computers, which caused more VR headsets emerging and increase in the interest for new research using the technology.

Locomotion is a key element of a VR application. It has been studied a lot and there is wide variety of locomotion techniques developed. In their review of different locomotion techniques [23] Boletsis proposes a topology that divides the techniques into four distinct VR locomotion types: motion-based, room scale-based, controller-based and teleportation-based. Motion-based and room scale-based locomotion types both rely on the physical movement. Room scale-based types map the users movements one-to-one into the virtual world, whereas motion-based types can use any kind of physical motion to enable the interaction. Controller-based locomotion types uses input devices, for example joysticks, to move the user around continuously. In teleportation-based locomotion types the movement is not continuous as the user is teleported instantaneously between positions.

VR sickness has been widely researched topic. The three major theories for the VR sickness introduced by LaViola [16] are:

- **Sensory Conflict Theory** explains the cause of the sickness as conflict between visual sensors, and the body motion detected by the vestibular organs. When a person is being moved around in the virtual environment while they are standing still in real world they experience *vection*, which is an illusion of motion caused by visual stimuli, when there is no movement in reality [24].

- **The Poison Theory** explains the cause of the sickness as body misreading the information it gets from visual and vestibular sensors, and thinks it has been poisoned. When the body thinks it has ingested some kind of toxin, it tries to remove it by causing nausea and vomiting.

- **The Postural Instability Theory** suggests that maintaining postural stability being one of the primary goals in humans, prolonged postural instability experienced while using VR causes motion sickness in the user.

The severity, and symptoms of the vr sickness being different for everyone makes it hard to determine exact reasons for it. It is also hard to measure exactly how much VR sickness someone is experiencing. There is questionnaires, such as Simulator Sickness Questionnaire (SSQ) [25], where the participant is asked to estimate how much of different symptoms they are experiencing, but also monitoring of physiological changes, such as heart rate, EEG, and stomach upset, has been used [26].

2.3. Shared Control

Shared control is often achieved by blending trajectories or policies (for example, [27]), using the autonomous controller as a safeguard for detecting collisions, or simply switching between autonomous and manual modes [28, 29].

In trajectory blending, commands coming from the autonomous controller are blended with the trajectory indicated by the user [30] [31] [32] [33]. As an example, Pappas et al. blend the trajectories using following function:

$$U_f = \alpha(\cdot)U_h + (1 - \alpha(\cdot))U_r, \tag{1}$$

where the U_h and U_r are the two trajectories to be blended, and the $\alpha(\cdot)$ is an arbitrator function which gives values between 0.0 and 1.0 and determines how much weight each of the trajectories will have in the final trajectory. However, apart from some of these ([30] [33]), there is no proper collision detection for the resulting trajectory.

In contrast, using the autonomous controller as a safeguard means that normally user has full control of the robot, but if some hazard or obstacle is detected by the robot it will take over the control preventing collision. This method was already used in lunar rovers [34] and has been used in many mobile robots afterwards [35][36][37]. This method protects the robot from obstacles, but actively controlling a robot for long distances can get tiring for the user.

2.4. Planning Methods

Finding the best path from one place to another has been for a long time, and there has been multiple different methods developed trying to solve the problem. We will call these methods Global Planners from now on. Edsger Dijkstra introduced **Dijkstra's Algorithm** in 1959 [38], which finds minimum total length path from the initial node to any other nodes in the system. It works by picking node adjacent to already found nodes with the lowest path length, and checking its neighboring nodes. If the distance to the neighboring tile through the current node is smaller than the one currently assigned to that node, it is replaced with the new lowest distance. After all the neighbors have been checked the current node is marked as visited and new node is chosen as a current node. The algorithm can be used to map distances from the initial node to all other nodes, or the algorithm can be stopped when the shortest distance to some specific node has been found.

Dijkstra's Algorithm is known to be computationally intensive and not that effective in finding path between two nodes. To solve this problem Hart, Nilsson and Raphael introduced **A* Algorithm** in 1968 [39]. In order to decrease the amount of nodes to be checked by the algorithm A* introduces an evaluation function

$$\hat{f}(n) = \hat{g} + \hat{h}, \tag{2}$$

in which $\hat{g}(n)$ is the cost of the path from the initial node to node n, and $\hat{g}(n)$ is the estimate of the cost of the optimal path from the node n to the goal node. This evaluation function is used to calculate cost for the new neighboring nodes of the

current node. As the nodes that are closer to the goal node will get lower cost given by the evaluation function, the algorithm expands the nodes that lead closer to the goal first thus decreasing the required computation.

Rapidly-Exploring Random Tree (RRT) was introduced by LaValle in 1998 [40]. It is not constrained into a grid like Dijkstra's Algorithm or A*. It starts with an initial vertex x_{init} and the goal region X_{goal}. In each iteration a random state is selected, and closest existing vertex is found. An input is selected that minimizes the distance between the selected random state and the closest vertex. This input is then checked for collisions, and the new state is only accepted if collision does not happen. The state is then added as a new vertex and the shortest path between the new and its closest vertex is added as an edge. This is then repeated until the goal region is reached.

2.5. Controllers

After we get the path from the robot to the goal, we need to get the robot to follow this path. This is done by the controller, or a Local Planner. **DWA** is a local planner introduced in 1997 [8]. It works in the velocity space, which is reduced by taking the robots dynamics and the obstacle region into account; velocities not reachable under the dynamic constraints, and velocities that would lead into an obstacle are discarded. From the remaining velocities a velocity that maximizes the objective function is chosen and sent to the robot to be executed. The objective function takes into account robots target heading, clearance and its velocity as follows:

$$G(v, \omega) = \sigma(\alpha\text{heading}(v, \omega) + \beta\text{dist}(v, \omega) + \gamma\text{velocity}(v, \omega)), \tag{3}$$

where v and ω are the set of velocities, and σ, β, and γ are scale factors for each of the parts of the objective function.

In this study we used improved version of the DWA included in the Robot Operating System (ROS) navigation package called **DWB** [41] which is an implementation of David Lu's DWB local planner [42]. It differs from the DWA by adding-plugin based critics and trajectory generation techniques. It also used a cost function for ranking the resulted trajectories instead of objective function. This means that the planner chooses the trajectory that minimizes the sum of all the critic functions. The planner is explained in more detail in section 3

Virtual Vector field was introduced by Borenstein and Koren in 1989 [43]. It is a controller that works by applying virtual *"forces"* to the robot from the outside. The obstacles are stored in an occupancy grid, and each occupied grid cell around the robot applies a force dependant on the distance from the robot, repelling the robot away from them. To make the robot navigate towards the goal there is also constant-magnitude attracting force pulling the robot toward the target point

2.6. Motivation

When designing control methods for an immersive telepresence robot, the user experience needs to be considered much more than with normal telepresence robots.

Completely manual control can get tiring after long distances and is more prone to human errors. The severity and causes of VR sickness also vary from person to person, which makes it hard to develop one single solution for global path that works for all. There is a need for system that allows each user to adjust the robots path to their own liking, without losing the safety of obstacle avoidance, and the goal of this thesis is to create one.

3. PROPOSED SHARED CONTROL METHOD

In this chapter we introduce the proposed method. First we describe the kinematic model of the robot and the underlying motion planning framework the proposed method is based on. Finally we describe the implementation of the proposed method.

3.1. Robot Model

This section explains the robot model we used in this implementation and experiment. This model was chosen as we have a physical robot with these specifications. You can see a picture of the physical robot in Fig. 1

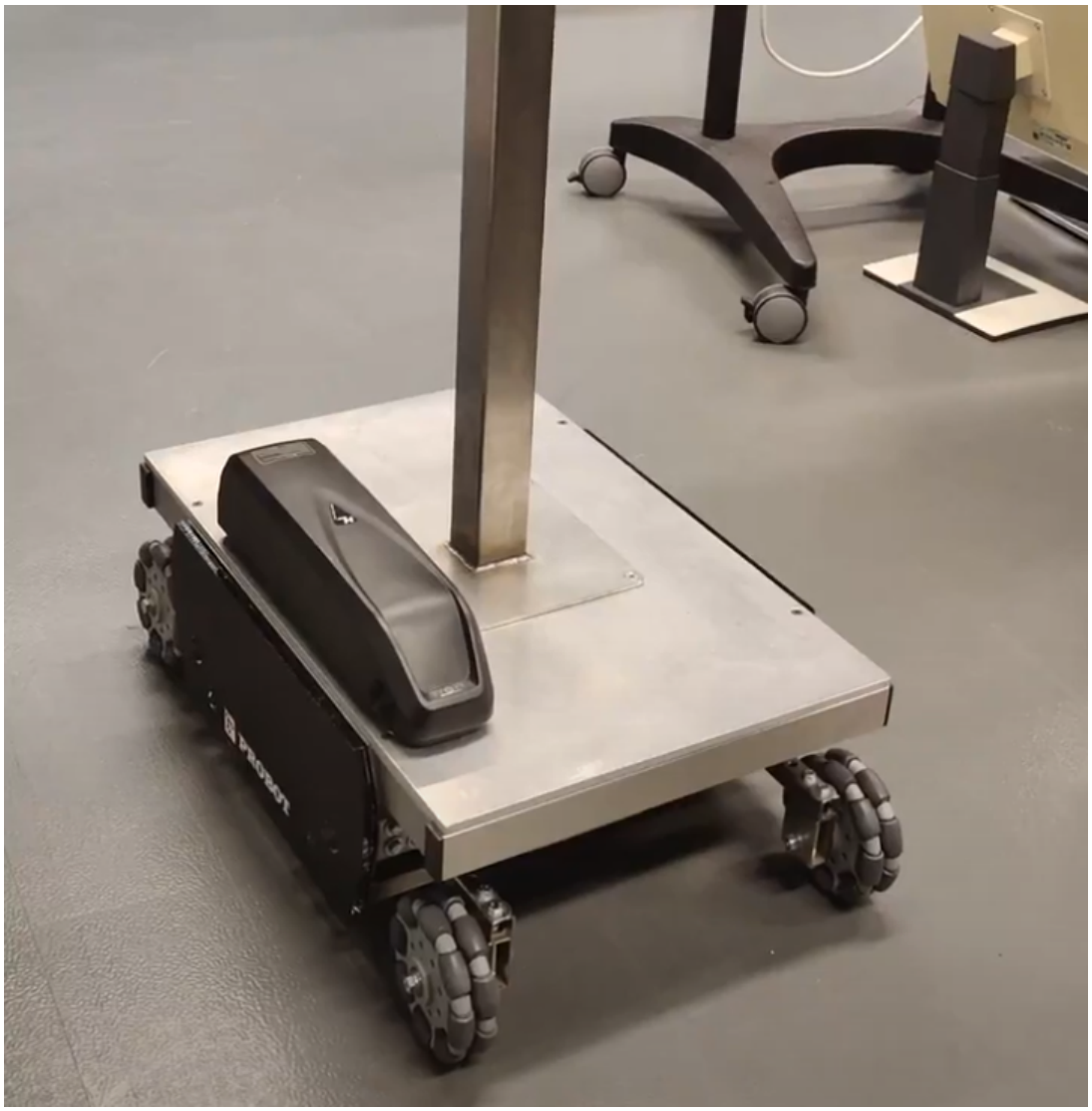


Figure 1. Picture of the physical robot our model in this paper is based on

We consider a telepresence robot shown in Fig. 2 that is a differential drive robot comprising two driving wheels and four omnidirectional wheels for balance. The robot configuration is given by (x, y, θ), in which (x, y) is the robot position and θ is the

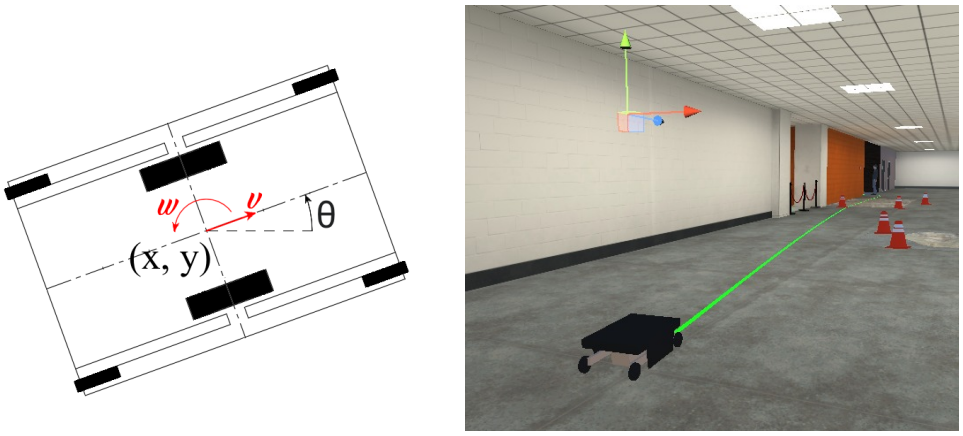
orientation with respect to a global reference frame, and it is constrained in the set $Q \subset \mathbb{R}^2 \times S^1$. The robot kinematic model is given by the equation:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega, \end{aligned} \tag{4}$$

in which the control input $u = (v, w)$ corresponds to the linear and angular velocities with respect to the robot-fixed reference frame (see Fig. 2a). The control input is mapped into target motor speeds using following equations:

$$\begin{aligned} \omega_{w1} &= \begin{cases} \dfrac{v}{r_w} & \text{if } \omega = 0 \\ \dfrac{(v + \omega\frac{l}{2})}{r_w} & \text{otherwise,} \end{cases} \\ \omega_{w2} &= \begin{cases} \dfrac{v}{r_w} & \text{if } \omega = 0 \\ \dfrac{(v - \omega\frac{l}{2})}{r_w} & \text{otherwise,} \end{cases} \end{aligned} \tag{5}$$

in which $(\omega_{w1}, \omega_{w2})$ is the target wheel rotational speeds, r_w is the radius of the wheel of the robot and l is the distance between the robots' wheels.



(a) (b)

Figure 2. (a) Diagram of the robot used in the experiment with state and control variables shown. (b) The simulated robot in virtual environment.

3.2. Motion Planning

Let $E \subset \mathbb{R}^2$ be a planar environment in which the robot is moving. There are obstacles, which are open subsets of E, that prohibit the robot from reaching certain configurations due to collisions. The obstacles change dynamically and this information is (locally) available to the robot. Therefore, let $E_{obs}(t) \subset E$ be the union of all the obstacles known to the robot at time t. The part of the planar environment that the robot can be in without collisions at time t is then denoted by

$E_{free}(t) = E \setminus E_{obs}(t)$. Let $\mathcal{A} : Q \rightarrow E$ be a mapping from the robot configuration to its footprint in the environment. The time-dependent free configuration space is defined as

$$Q_{free}(t) = \{q \in Q \mid \mathcal{A}(q) \in E_{free}(t)\} \tag{6}$$

Let $q_g = (x_g, y_g, \theta_g)$ and $q_0 = (x_0, y_0, \theta_0)$ be the goal and the initial configurations, respectively. Conventionally, the motion planning problem is defined as finding a control-trajectory $\tilde{u} : [0, T] \rightarrow U$ such that the state-trajectory $\tilde{q} : [0, T] \rightarrow Q$ computed as forward integrating Equation (4) starting from q_0 satisfies $\tilde{q}(t) \in Q_{free}(t), \forall t \in [0, T]$ and $\tilde{q}(T) = q_g$. Typically, a control-trajectory that is optimal with respect to a relevant metric is sought and the final time T is not fixed.

As common in the literature, we consider that the motion planning for the robot is achieved by two interacting modules: the *planner* and the *controller*. The planner is responsible for computing a length-optimal path to the goal (not necessarily feasible with respect to the robot kinematics), whereas the task of the controller is to compute the control input that tracks the output of the planner. We assume that a planner is in place and address the inclusion of the user input at the controller level. Before describing our problem, we will briefly explain the planner. Since the considered environment is dynamic, the planner is invoked at regular intervals t_k, $k = 0, \ldots, K$ to re-plan using the currently available information. Suppose the estimate of the robot configuration at t_k is available and denote it by $\hat{q}_k = (\hat{x}_k, \hat{y}_k, \hat{\theta}_k)$. Then, given the current environment $E_{free}(t_k)$ and $\hat{q}_k$, the output of the planner is a length-optimal path $\pi_k : [0, 1] \rightarrow E_{free}(t_k)$ such that $\pi_k(0) = (\hat{x}_k, \hat{y}_k)$ and $\pi_k(1) = (x_g, y_g)$. It is often referred to as the global path. Note that the computed path is not kinematically feasible[1] as it doesn't take into account the kinematic restrictions of the robot. However, it is assumed that the planner considers the robot size so that the path has sufficient clearance. Since the computational load of searching the whole position space is high, it is expected that the planner works at a low frequency. Hence, it is mainly the task of the controller to avoid obstacles.

We consider a controller in line with methods based on searching the control input space such as DWA [8] and trajectory rollout [44]. These methods search a limited set of admissible controls (velocities) with respect to the actuation constraints such that given the current linear and angular velocities of the robot, only the ones that are achievable within a short length of time are considered. Consequently, assuming that the controls will stay constant, the respective state-trajectories are evaluated with respect to an objective function. The objective function is selected so that it captures, for example, the obstacle clearance and vicinity to the global path. The control input corresponding to the best trajectory in terms of the cost determined by the objective function is then passed to the robot. This procedure is repeatedly executed with a frequency of $10hz$ for each t considering $Q_{free}(t)$ and π_k such that $t \in [t_k, t_{k+1})$. Due to the computationally infeasible exhaustive search over the set of admissible controls, typically only a set of sampled controls are passed to the trajectory evaluation step. Let $U^{\Delta}(t)$ be the set of sampled admissible controls at time t and let $\Delta\tau$ be the time window used to integrate the dynamics. The set of control pairs such that respective

[1] A planner that computes kinematically feasible paths can also be used. In that case, π maps to $Q_{free}(t_k)$ and satisfies Equation (4)

trajectories are collision free are denoted as $U^{\Delta}_{free}(t)$. This implies that for each element $u \in U^{\Delta}_{free}(t)$ the trajectory $\tilde{q} : [t, t+\Delta\tau] \rightarrow Q$ computed considering a constant control for $\Delta\tau$ satisfies $\tilde{q}(\tau) \in Q_{free}(t), \forall\tau \in [t, t+\Delta\tau]$. Then, the best control input to pass to the robot at time t is determined as

$$u^*_t = (v^*_t, w^*_t) = \underset{u_t \in U^{\Delta}_{free}(t)}{\arg\min} \; s_{nv} J_{nv}(u_t) \tag{7}$$

in which $J_{nv}(u) = (J_1(u), J_2(u), \ldots, J_d(u))^T$ is a d-dimensional vector of objective functions capturing the aspects relevant to path tracking, obstacle avoidance, and goal achievement and $s_{nv} = (s_1, s_2, \ldots, s_d)$ is the respective vector of weights.

3.3. Proposed Method

We address the problem of designing a controller that tracks the path computed by the planner taking into account the input given by the operator. This corresponds to enabling the operator to apply slight modifications to the trajectory executed by the robot without directly controlling the robot motion. Suppose that the operator can indicate a preference for robot motion which is then mapped to a control command $u_h = (v_h, w_h)$. We propose a DWA-based controller to incorporate the user input in the selection of best control input pair to pass to the robot. To this end, we augment the minimization problem in Equation (7) with an additional cost function that penalizes the difference between the selected control pair and the operator input. The cost function to evaluate the control input pairs and respective trajectories is described as

$$J(u_t) = \begin{cases} s_{nv} J_{nv}(u_t) + s_{sh} J_{sh}(u_t) & \text{if } \gamma(t) = 1 \\ s_{nv} J_{nv}(u_t) & \text{otherwise,} \end{cases} \tag{8}$$

in which (v_h, w_h) is the pair of control inputs given by the operator at time t and γ is a function that maps t to 1 if a user input is given and to 0 otherwise. The cost component resulting from the user input is defined as

$$J_{sh}(u_t) = \Big(|v_h - v_t|, |w_h - w_t| \Big)^T \tag{9}$$

and $s_{sh} = [s_v, s_w]$ is the vector of respective weights. Consequently, the best control input in $U^{\Delta}_{free}(t)$ is determined as

$$u^*_t = (v^*_t, w^*_t) = \underset{u_t \in U^{\Delta}_{free}(t)}{\arg\min} \; J(u_t), \tag{10}$$

We introduce a delay in the transition to the autonomous navigation to ensure that once the user input ceases, the resulting motion does not involve sudden rotations to compensate for the potential divergence from the global path due to user input. Let t be the moment when the operator stops interacting with the robot. Then, a constant pseudo-input is given to the controller such that v_h is the velocity command given at t and $w_h = 0$ for all $\tau \in [t, t+\delta]$, in which δ is the delay.

4. EXPERIMENT

In this chapter we describe the user study in detail. First it explains what software was used in the implementation. Then it introduces the three different controller types the participants used during the study. It also describes both the virtual environment used in the study, and the real life study setup. It introduces the hypotheses pre-registered for this study, goes through whole study procedure, explains what things we measured during the study and introduces the participants.

4.1. Tools Used

The hypotheses were tested using a simulated environment created using a game engine called Unity version 2020.3.12 Unity was chosen as it has full support for VR with a possibility for a realistic environment. We provide a more detailed description of the environment in section 4.4.

ROS is a modular framework that was originally designed for large-scale service robots [45]. It is multi-lingual, free, and open-source which makes it a really good candidate to be used in research. The navigation package [46] provides a good framework for autonomous navigation.

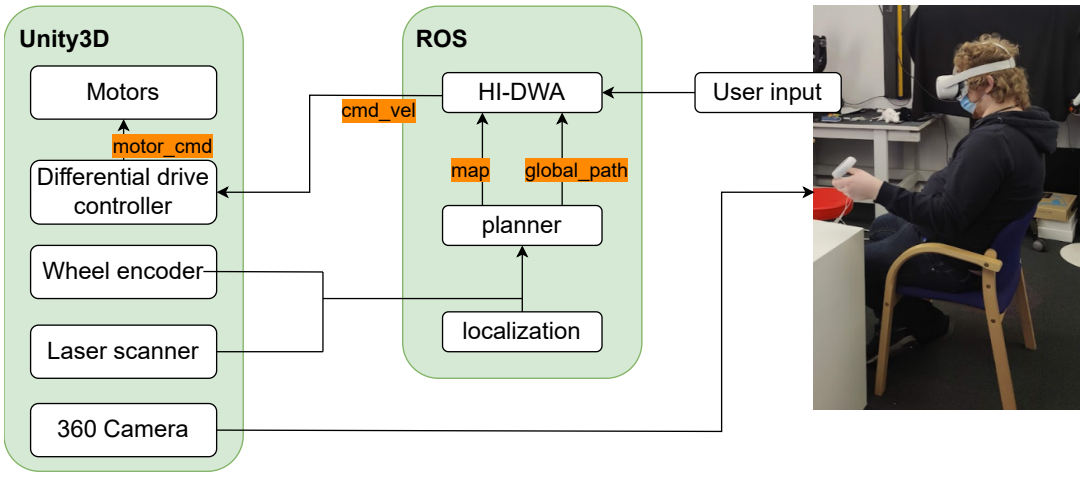


Figure 3. Diagram of the communications between the ROS2 and the simulated environment in Unity

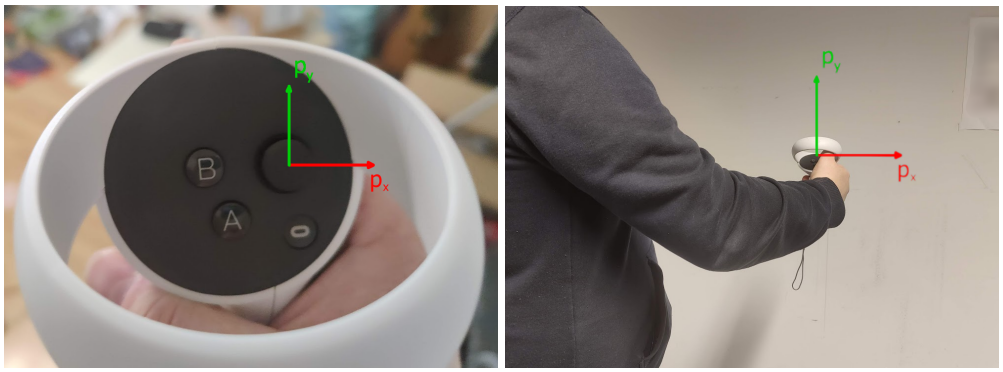
The navigation of the robot was implemented using the Nav2 project [47]. We use the default planner and localization plugins provided by Nav2 project. In particular, we use the default critics (objective functions forming the vector J_{nv}) and their default weights. Therefore,

$$\begin{aligned} J_{nv}(u_t) =& s_{pa}\texttt{PathAlign}(u_t) + s_{pd}\texttt{PathDist}(u_t) \\ &+ s_{bo}\texttt{BaseObstacle} + s_{ga}\texttt{GoalAlign}(u_t) \\ &+ s_{gd}\texttt{GoalDist}(u_t) + s_{rg}\texttt{RotateToGoal}, \end{aligned}$$

in which for each trajectory corresponding to the numerical integration of the control u_t for $\Delta\tau$, `PathAlign` ($s_{pa} = 32{,}0$) and `PathDist` ($s_{pd} = 32{,}0$) penalize the trajectory based on the distance from the global path and how well it is aligned to it, respectively. Similarly, `GoalAlign` ($s_{ga} = 24{,}0$) scores a trajectory based on how well the robot aligns with the goal pose and `GoalDist` ($s_{gd} = 24{,}0$) scores it based on how close the trajectory gets the robot to the goal pose. `BaseObstacle` ($s_{bo} = 0{,}02$) scores a trajectory based on its distance from the obstacles. Finally it includes also a binary critic named `Oscillation` that prevents backwards-forwards motion by penalizing such trajectories with infinite cost. We refer the reader to Nav2 documentation [48] for more detailed explanations.

The communication between the Unity program and the ROS2 was done using the official ROS-TCP-Connector [49], which consists of a plugin for both ROS2 and Unity. These plugins are able to communicate with each other and transfer ROS2 messages between the ROS2 nodes and Unity. The communications between ROS2 and Unity are presented in Fig. 3

4.2. Controllers



(a) (b)

Figure 4. (a) Joystick coordinates (p_x, p_y) related to the physical joystick position (b) $p_x - p_y$ plane related to the hand position

Switching (SW)

Switching control refers to handing direct control of the robot to the operator when the operator wants to alter the course of the robot motion. Switching control systems that are similar to the setup considered in this thesis have been proposed for instance in [28] and implemented in the commercial Double 3 telepresence robot. For the rest of the time the robot is moving autonomously. The control input to send to the robot at time t, is given by

$$u_t^* = \begin{cases} u_h = (v_h, w_h) & \text{if } \gamma(t) = 1 \\ \underset{u_t \in U^{\Delta}_{free}(t)}{\arg\min} \; s_{nv} J_{nv}(u_t) & \text{otherwise,} \end{cases} \tag{11}$$

To make the switching explicit, we implemented a button such that once it is pressed the control is transferred to the operator. If after pressing the button no user input is given, the respective $u_h = (v_h, w_h) = (0, 0)$ and the robot stops. The user input is given using the joystick of an Oculus Quest 2 controller. Let $(p_x, p_y) \in [-1, 1] \times [-1, 1]$ be the joystick coordinate corresponding to the user input, with $(0, 0)$ indicating the neutral (center) position, see Fig. 4 a). This coordinate is mapped to $u_h = (v_h, w_h)$ as follows:

$$v_h = \begin{cases} 0 & \text{if } |p_y| \leq 0.1 \\ p_y^2 \operatorname{sgn}(p_y) v_{max} & \text{otherwise,} \end{cases} \tag{12}$$

and

$$w_h = \begin{cases} 0 & \text{if } |p_x| \leq 0.1 \\ p_x^2 \operatorname{sgn}(p_x) w_{max} & \text{otherwise,} \end{cases} \tag{13}$$

Shared Control-Joystick (SJ)

In this study we considered two implementations of the shared control method, introduced in Section3. The first of these is the SJ method. Similar to Switching Control, also in this case, the operator uses a joystick to provide input to the system. To keep controlling the robot simpler, we allow the user only to affect the rotational speed but not the linear speed. Therefore, for all u_h, v_h is set to maximum velocity to avoid prioritizing controls corresponding to rotate in place and w_h is computed using Equation (13). The control input u_t^* passed to the robot at time t is determined using Equation (10). To find a good combination of the relative weights, that is, $s_{sh} = (s_v, s_w)$ for penalizing the costs $J_{sh}(u_t)$ that take part in $J(u_t)$ and to determine a sufficient delay δ, we ran a pilot test ($N = 8$). The participants tried this control method with four different parameter combinations, resulting in four conditions. For (s_v, s_w, δ) we tested the following four combinations: $(200, 400, 2s)$, $(400, 800, 2s)$, $(200, 400, 1s)$, $(400, 800, 1s)$. When asked which condition they felt was the best, 6 out of 8 participants picked condition 2, one participant picked conditions 3 and 4 as equally good, and one participant picked conditions 1 and 2 as equally good. Since condition 2 was selected by the majority of the users we used the parameters used in condition 2 for the main study ($s_{sh} = (s_v, s_w) = (400, 800)$ and $\delta = 2s$).

Shared Control-Gesture (SG)

This method differs from SJ only in terms of the way the input is received from the operator. In SG, the input consists of a gesture control and a button to indicate the trajectory. When the operator presses down the button, we obtain the current position and orientation of the hand from the tracking system of the Oculus Quest 2 controller. Then this is mapped to coordinates in the $x - y$ plane shown in Fig. 4 b). The initial position of the hand is set as the origin of this plane and the normalized horizontal movement of the hand is mapped to w_h similar to Equation (13). Just like with the SJ method, for all u_h, v_h is set to maximum velocity.

Figure 5. The physical setup of the study with a participant

4.3. Physical Environment

The experiments were ran in a closed room, participants sitting down on a stationary chair. To avoid getting the HMDs cables stuck anywhere, a pulley system was used to hang the cable from the ceiling. You can see the physical setup of the study in Fig. 5

Three laptops were used during the experiments. Two of them were operated by the experimenter, one for running the VR application and one for running the ROS nodes. To comply with the COVID-19 restrictions these laptops were located two meters away from the participant and the experimenter only got close to the participants if it was needed. The third laptop was used by the participants to fill out the questionnaires after each task.

4.4. Virtual Environment

The virtual environment was built around a 3D reconstruction of a part of the University of Oulu campus called Tellus. However, this environment was altered and corridors and obstacles were added in such a way that even though the participants

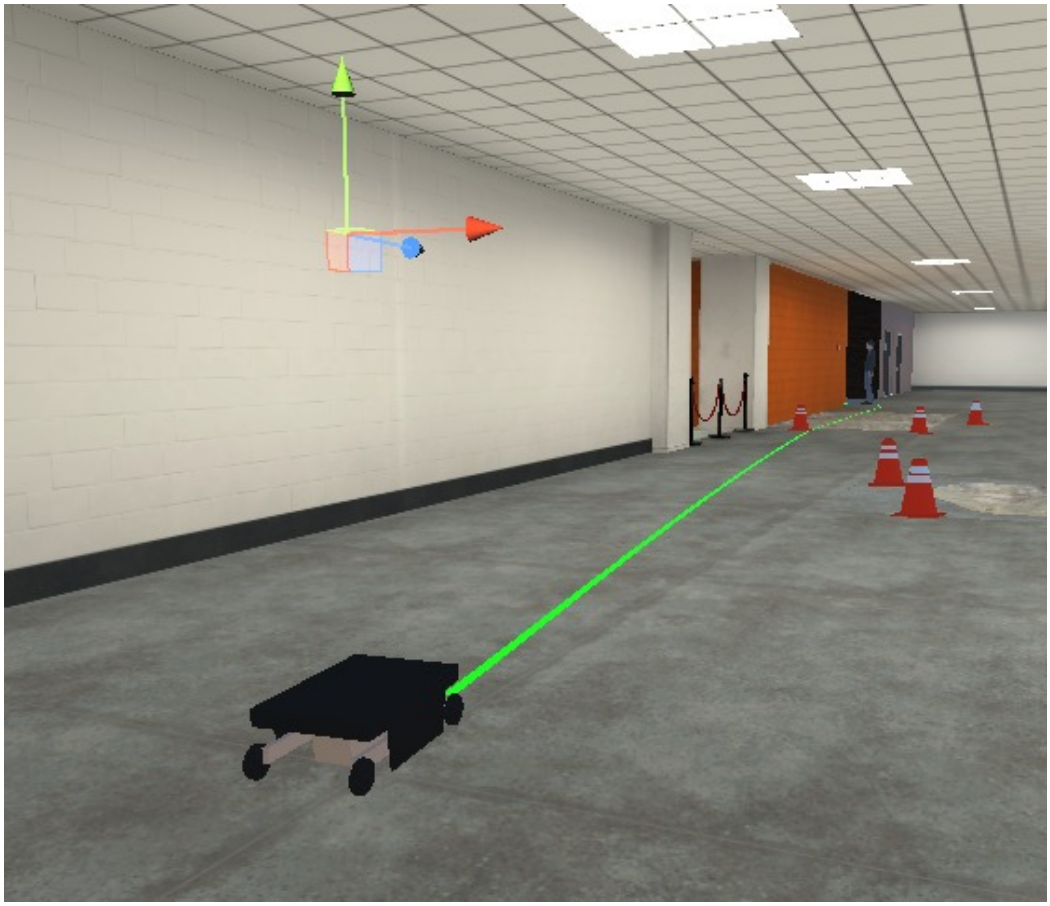


Figure 6. View of the environment with the robot.

might recognize the Tellus environment, they would not be able to use that to their advantage. The corridors were created by using assets from the Unity Asset Store.

To give the participants a reason to alter the robot's path we included special *regions to avoid* that consisted of potholes, scaffolding, cardboard boxes, traffic cones, and bumpy areas. These were not visible to the robot's sensors and were not marked in the map so that they could not be avoided by the autonomous controller. However, it was not impossible for the robot to pass through these obstacles. The regions to avoid were same for every task, but we used two sets of visible obstacles such as the cardboard boxes or scaffolding so that the participants would not be able to predict the exact position of the obstacles based on their previous tasks. Examples of the different obstacles in the same position can be seen in the Fig. 7.

The route of the robot was also designed in a way that it would encourage participants to alter the path of the robot. For example, the robot uses a path that takes the user close to the corners, and goes through a group of people. Getting close to corners has been shown to make people uncomfortable [19], and walking through a group of people when other paths are available is seen as rude behavior.

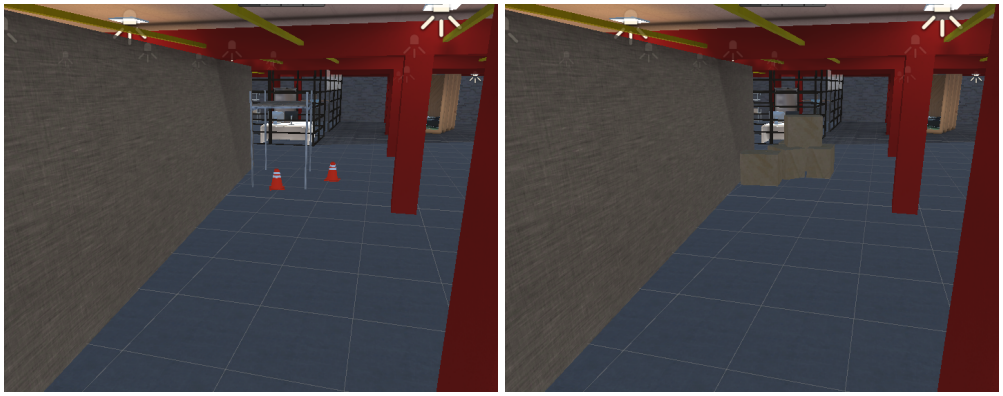


Figure 7. Two different sets of obstacles used in the same location.

4.5. Hypotheses

In a scenario of a semi-autonomous telepresence robot navigating an environment in which users immersed in the robot through a head-mounted display (HMD) can alter the path executed by the robot either by switching to manual guidance or by indicating the direction that they would like to go, we will test the following hypotheses:

H1: SJ condition is preferred as indicated by asking directly which condition was preferred.

H2: Perceived workload is lower under SJ as indicated by asking directly which condition was easier and NASA Task Load Index (NASA-TLX) questionnaire after each condition.

We pre-registered the hypotheses, with the procedure and analyses to be used in the study, in Open Science Foundation (OSF) `https://osf.io/q9ubx`.

4.6. Procedure

Before the participant enters the room the experimenter prepares the questionnaires by filling out the participant's ID and other initial information. The ID is determined by the order number of the participant and the scenario the participant would experience first. The participants were presented with three conditions corresponding to the SW, SJ, and SG control methods. The conditions SW and SJ, and the obstacle sets were presented in counterbalanced order such that any of the four combinations was seen first an equal amount of times by the participants. The SG method was always presented last due to its experimental nature, and it always had different obstacles than the SJ method, as these two methods would be compared to each other. The ID had the order number in front, then either SJ or SW depending on if the first scenario would be SJ or SW respectively. The end of the ID would be either 1 or 2 depending on the used obstacle group. As an example a participant who would be fifth in the order, had SJ as their first control method with the obstacle group of 2, would get ID of 05SJ2.

Upon arrival, the participants were greeted by an experimenter, asked to sit down, and signed a form to indicate their consent to participate. Next, the experimenter asked

the participants if they were feeling nauseous or had a headache in an effort to pre-screen people feeling sick already before the experiment began. Then, the participants were told the general instructions by the experimenter, and shown how to put on the HMD.

After the experimenter made sure that the participant knew how to put on the headset properly, they read out the instructions for the first task. The participants were told that they needed to get to a meeting that they were late to using a telepresence robot and that there were potholes, cardboard boxes, traffic cones, and scaffolding in the way of the path which are not visible to the robot's sensors. Despite not being explicitly told to avoid those regions, they were encouraged to do so by saying that they could get slowed down or feel uncomfortable (bumpy tiles) if they did not. They were also told that the path the robot would try to follow was shown as green line on the floor.

Before each task, the participant was asked to practice the controls in a short practice environment. After the practice, the participants were asked if they were ready for the task, and the proper task scenario was started by the experimenter. After the task, the participants were asked to remove the HMD and fill out a questionnaire regarding their experience with that specific control method. The same procedure was repeated two more times with the other control methods. Finally, the participants were given 20€ Amazon vouchers for participating. After each experiment, all the equipment was disinfected as a precaution regarding COVID-19. Precautions were also taken during the experiment by using disposable face covers with the HMD, the experimenter always wearing a mask, and the experimenter keeping a safe distance from the participant except if help was needed.

4.7. Measures

During the experiment we collected data by questionnaires after each of the tasks and by logging data received from the HMD and the virtual environment.

To see how the robot was moving, we recorded the position and orientation of the robot through each of the tasks. We also recorded time stamps for the moments when the participant used their controller. This was done to determine when the robot was moved by the participant and when it was moving autonomously. An example of the tracked path with the time stamps can be seen in Fig. 8. We also measured the head movements of the participants using the tracking system of Oculus Quest 2 to see where the participants were looking at while performing the task. The tracking was done by obtaining the positions and orientations of the robot and the head of the participant every frame. These were then stored in a file together with corresponding time stamps. The robot was tracked only in the global coordinate frame, whereas the head was tracked both in global coordinate frame, and in the robot's coordinate frame.

Our program also measured the time taken to finish the task, and counted how many times the participant collided with the obstacles. The obstacles seen by the robot's sensors were counted separately from the obstacles not visible to the robot.

At the beginning of each questionnaire, we asked the participants to fill out a SSQ [25]. It is a questionnaire often used to measure the sickness that results from using VR, and consists of questions regarding 16 different sickness symptoms that are scored based on severity of experience. Weighted scores are used to calculate the total

Figure 8. An example path taken by the robot. The green line shows the robots autonomous movement, and the red line shows the robots movement when it is controlled by the user.

score for the sickness. Higher scores indicate greater levels of sickness experienced. Participants were also asked to fill out a NASA Task Load Index questionnaire, which is used to measure six dimensions of workload (mental demand, physical demand, temporal demand, performance, effort, and frustration) of the task. Each dimension is compared between the methods individually to find if one of the methods is perceived as more demanding than the others. These questions were followed by a forced-choice question if there was any point where the participant wanted to alter the path but did not, and 7-point Likert-scale questions about the participant's perceived control over the robot, the ease of altering the path, and their comfort while altering the path.

After the second task, we additionally asked forced-choice questions about the participants' preference between the two methods, and a comparison between the two methods on their control and ease of use. We also asked open-ended questions about the reasons for their choices and if there were any specific situations where they would prefer one of the methods over another.

Finally, after the third task, we asked forced-choice questions about whether the SG method made them feel more as if they were there in the environment or if they felt more in control of the robot than in the previous tasks. We also asked their preference over all three different methods and open-ended questions about reasons for their choices. In the end we asked about participant's VR and gaming experience and their demographics.

4.8. Participants

The study participants were recruited from the University of Oulu campus and community. We aimed to have 32 participants , but due to the exclusion of four people from the study, we ended up running 36 total. Three of the excluded participants quit the experiment due to sickness symptoms, and one was excluded for not following the instructions and moving the robot outside of the intended environment, which caused them to get stuck and thus they were not able to finish their task without a reset. Of the 32 included participants, 19 were men and 13 were women. 18 of the participants reported that they play games more than once or twice a month, but only four said that they use VR more than once or twice a month. 17 participants wore glasses or contacts and one was colorblind.

5. RESULTS AND DISCUSSION

In this chapter we present the results of the study.

5.1. Results

All tests were run in SPSS with significance levels set to $0{,}05$ and with a 95% confidence interval.

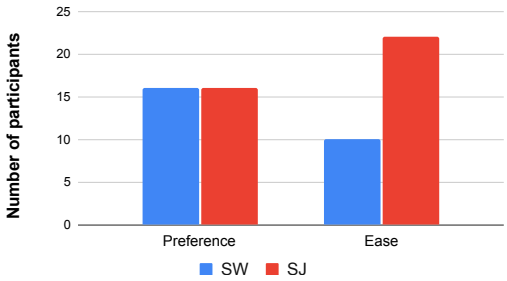


Figure 9. The distributions of responses to the questions regarding preference, ease

5.1.1. Confirmatory Analysis

Both methods were preferred equally (H1 rejected) Fig. 9 shows the distributions of the responses given by the participants to the forced-choice questions regarding preference and ease of use, and comfort. When asked “Which control method did you prefer?" 16 out of 32 participants (50%) selected the SJ condition showing no tendency in either direction in preference.

Perceived workload was lower under SJ (H2 confirmed) When asked “Which control method was easier to use?" 22 out of 32 participants $(68{,}75\%)$ selected the SJ condition. An exact binomial test with exact Clopper-Pearson 95% CI was performed, showing that the condition SJ was found significantly easier compared to the SW condition, $p = 0{,}026$ (one-sided) and had a 95% CI of $50{,}0\%$ to $83{,}9\%$.

SJ required significantly less effort when compared to SW (H2 confirmed) We compared the differences between the TLX-scores across conditions for each dimension of workload and found statistically significant difference only in the effort dimension. A Wilcoxon Signed-Ranks test is performed to compare the TLX effort scores for SJ ($Mdn = 15{,}0$) and SW ($Mdn = 25{,}0$) (see Fig. 10b for the score distributions). The test indicated that SJ elicited statistically significantly lower effort scores compared to SW, $Z = -1{,}687$, $p = {,}047$, $r = 0{,}30$ (one-sided).

5.1.2. Exploratory Analysis

In addition to the confirmatory analyses, we performed exploratory analyses to interpret the results better.

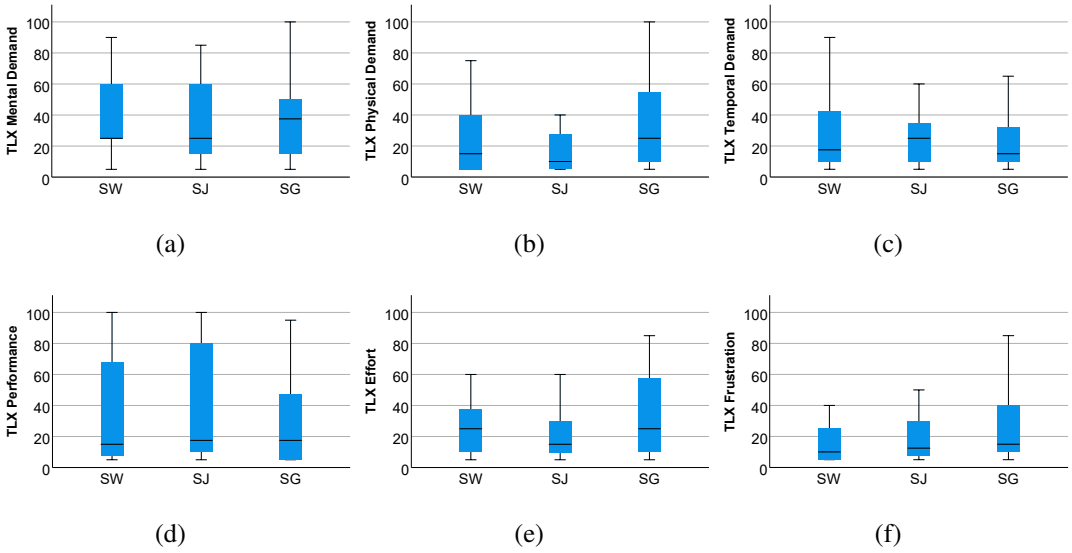


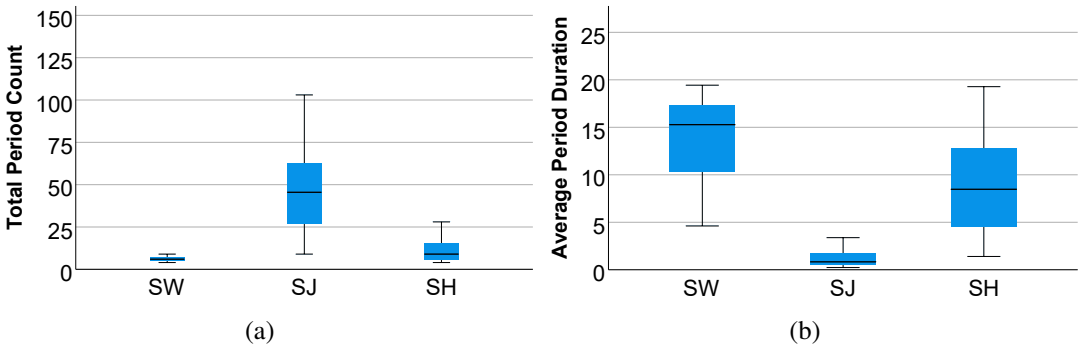
Figure 10. Comparison of the TLX scores (a) mental demand (b) physical demand (c) temporal demand (d) performance (e) effort (f) frustration.

Quantitative Analyses

Using SJ elicited lower task completion times The task completion time data was analyzed to see if one method would result in faster task completion. The respective distributions for SJ and SW conditions did not follow a normal distribution as indicated by a Shapiro-Wilk test, $W = 0{,}773$, $p < 0{,}001$ and $W = 0{,}852$, $p < 0{,}001$, respectively. Therefore, a Wilcoxon Signed-Ranks test (two-sided) was performed to compare the differences between the task completion times for SJ ($Mean = 172{,}37s$) and SW ($Mean = 187{,}17s$) conditions. The test indicated that SJ elicited significantly faster task completion compared to SW, $Z = -4{,}207$, $p =< 0{,}001$, $r = 0{,}74$.

No difference in number of collisions To see if one control method helped to avoid *regions to avoid* more, we analyzed the number of regions that the path executed by the robot intersected with. Comparing the number of regions not avoided for SJ ($Mean = 0{,}219$) with SW ($Mean = 0{,}406$) we did not observe any significant difference as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -1{,}281$, $p = 0{,}213$, $r = 0{,}226$. Considering the obstacles, in total, we observed three collisions under SW condition. There were no collisions for SJ as it is inherently collision-free.

SJ was used significantly more often and in shorter segments During the tasks we recorded time stamps when the participants started and ceased giving the user input to the robot. To compare the number times each control method is invoked, we compared the number of such intervals (each interval starts with the user input and ends when it ceases) across conditions. The SJ method was used significantly higher number of times ($Mean = 47{,}4$) than the SW method ($Mean = 5{,}9$), as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -4{,}937$, $p =< 0{,}001$, $r = 0{,}873$. It was also used significantly more than the SG method ($Mean = 11{,}5$), as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -4{,}873$, $p =< 0{,}001$, $r = 0{,}861$. Similarly, the SJ method was used in shorter segments (as measured by the average duration of each interval) ($Mean = 1{,}698s$) when compared to either the SW method ($Mean = 28{,}103s$) or the SG method ($Mean = 9{,}688s$), as indicated by the



(a) (b)

Figure 11. Comparison of the (a) total counts of use, and (b) average duration of the input commands, for each of the methods

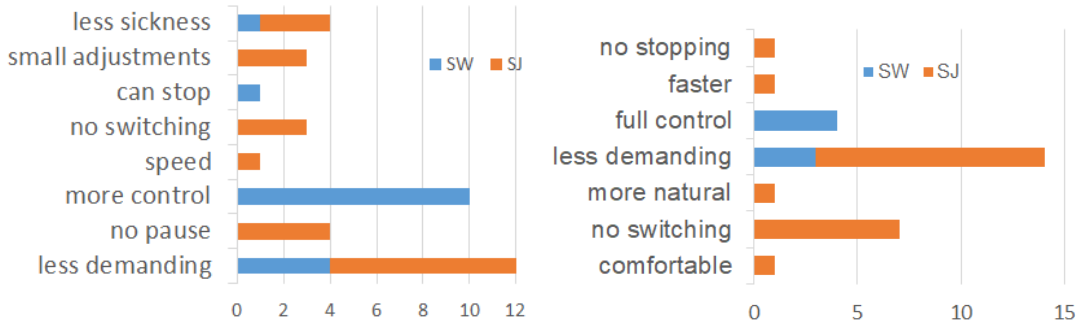
Wilcoxon Signed-Ranks test (two-sided), $Z = -4{,}937$, $p =< 0{,}001$, $r = 0{,}873$, and $Z = -4{,}806$, $p =< 0{,}001$, $r = 0{,}0{,}850$ respectively. The distributions of the values can be seen in Fig. 11.

There was no difference in sickness scores We tested whether one condition induced more sickness. Comparing the weighted total SSQ scores across the conditions SJ ($Mean = 44{,}5294$) and SW ($Mean = 48{,}3863$), we did not observe a significant difference between the scores, as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -0{,}809$, $p = 0{,}427$, $r = 0{,}14$. However, we observed a carryover effect on sickness with 15 minutes breaks; total weighted SSQ scores after the task 2 ($Mean = 54{,}1131$) was significantly higher compared to the task 1 ($Mean = 38{,}80$), as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -3{,}648$, $p =< {,}001$, $r = 0{,}64$.

Participants who preferred SJ found it slightly easier To find out the reasons for participants' choices for their preference between the SW and SJ methods, we filtered the participants into two groups based on their preference. We then compared the Likert-scale ratings for the easiness and the feeling of control over the robot within those groups.

From the participants who preferred the SW method, when comparing their ratings of the easiness of the method, there was no significant difference between the SW method ($Mean = 2,0625$) and the SJ method ($Mean = 2{,}5625$) as seen from a Wilcoxon Signed-Ranks test (two-sided), $Z = -1{,}144$, $p = 0{,}253$, $r = 0{,}202$. However, from the participants who preferred the SJ method, we found, at best, weak evidence that the easiness score was lower with the SJ method ($Mean = 1{,}6250$) than with the SW method ($Mean = 2{,}3125$), $Z = -1{,}942$, $p = 0{,}052$, $r = 0{,}343$. Similarly, of the participants who preferred the SW method, only 10 out of 16 ($62{,}5\%$) said that it also felt easier, while all 16 participants who preferred the SJ method also picked it as easier method.

Participants who preferred the SW said having more control over the robot From the participants who preferred the SW method, when comparing their ratings of the control over the robot, we found out that there is, at best, weak evidence that the SW method ($Mean = 1{,}9375$) had more control over the robot than the SJ method ($Mean = 2{,}8750$), as seen from a Wilcoxon Signed-Ranks test (two-sided), $Z = -1{,}943$, $p = 0{,}052$, $r = 0{,}343$. From the people who preferred the SJ method, a



(a) (b)

Figure 12. Frequently found codes for questions (a) “Please explain why you prefer that control method”. (b) “Please explain why that control method felt easiest to use”

Wilcoxon Signed-Ranks test indicated that there was no significant difference between the the means of the control over the robot ratings between the SW method ($Mean = 2{,}3750$) and the SJ method ($Mean = 2{,}4375$), $Z = -0{,}265$, $p = 0{,}791$, $r = 0{,}047$.

SG was found significantly harder and more physically demanding than SJ and SW, but it was preferred by almost half of the participants With our exploratory third control method, we wanted to see if participants would prefer using their gestures (hand) to give inputs to the shared controller instead of the joystick. When we asked "Which of the three methods did you prefer?", the SG method was preferred by 15 out of 32 participants ($46{,}9\%$) while the SW method was preferred by only 8 ($25{,}0\%$) and the SJ method was preferred by only 9 participants ($28,1\%$). When asked "How easy was it to alter the path of the robot", participants felt that the SJ method was significantly easier to use ($Mean = 2{,}0937$) than the SG method ($Mean = 2{,}9687$), as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -2{,}381$, $p = 0{,}015$, $r = 0{,}421$. When comparing the physical demand score between the SJ method ($Mean = 3{,}969$) and the SG method ($Mean = 7{,}031$), the SG method had significantly higher score, as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -2{,}785$, $p = 0{,}004$, $r = 0{,}492$. Similarly the participants felt that the SG method required more effort ($Mean = 6{,}969$) than the SJ method ($Mean = 4{,}875$), as indicated by a Wilcoxon Signed-Ranks test (two-sided), $Z = -2{,}034$, $p = 0{,}041$, $r = 0{,}360$. Other TLX scores did not have significant differences. TLX score distributions can be seen in Fig. 10.

Qualitative Analysis

The open-ended data was analyzed using the thematic analysis method with inductive approach [50]. The responses to the open-ended questions regarding why the control method felt easiest to use the greatest number of comments (44%) said that it was *less demanding*, followed by *no switching* (22%) and *full control* (13%). See Fig. 12b for the frequently found keywords in the participants’ responses.

Participants reported preferring the SJ because of its easiness, and the SW for percieved control over the robot Fig. 12a shows the frequently found codes in the responses to the open-ended question asking why participants preferred the control method, divided by which method was preferred. The most frequently found code was *less demanding*. Eight (50%) participants who preferred SJ and four (25%) participants who preferred SW found these methods less demanding (“*It felt that it would be less*

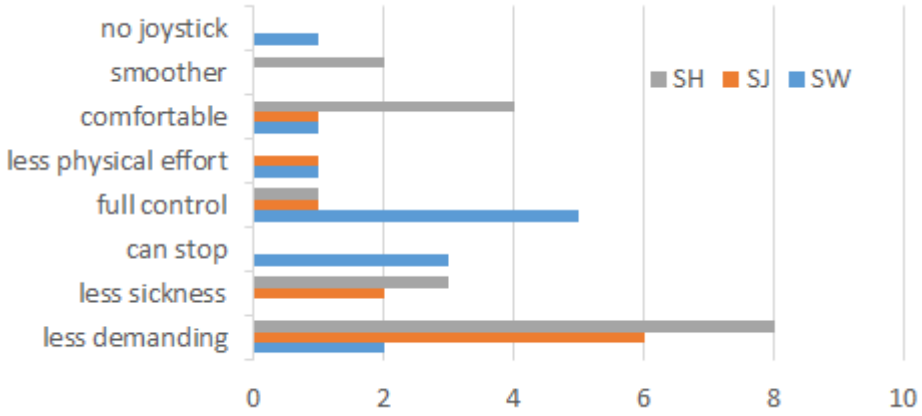


Figure 13. Frequently found codes for question “Please explain why you prefer that control method” asked after the third, experimental, method

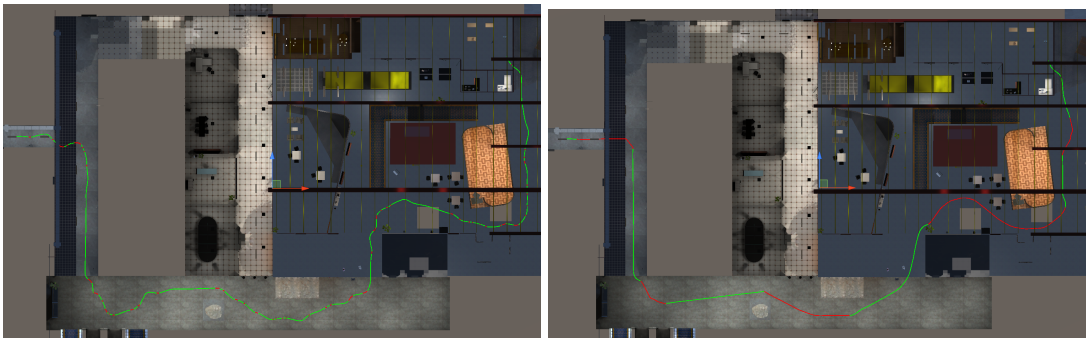
laborious to just make small adjustments to the path when needed rather than drive manually"). The biggest factor for participants who preferred SW was having *more control* that is found in the responses of 10 out of 16 participants (“*With the first method it was always clear who was in control. The second method felt like a constant struggle.*").

SG was reported being more comfortable by the participants Fig. 13 shows the frequently found codes in the responses to the open-ended question asking why participants preferred the control method after they had tried the experimental SG method. The same responses were found in these questions too. Five (56%) participants who preferred SW mentioned having full control over the robot. Six (75%) of the participants who preferred SJ, eight (53%) of people who preferred SG, and two (22%) found these methods less demanding. Another big factor for participants who preferred SG was it being *comfortable* that is found in the responses of 4 out of 15 participants (“*By far the most comfortable and low effort.*").

5.2. Discussion

Even though this study failed to prove any of the control methods being preferred significantly more than others, it brings interesting insight into why the participants preferred different methods. Participants found SJ statistically significantly easier than SW. SJ was also reported to be less demanding compared to SW.

One reason why participants preferred SW but found SJ easier was the *feeling of control*. For example, one participant preferred SW because *“You can have full control and navigate the path from the location of your choosing"*, but found shared control easier because *“its easier because you can relax and see where you want to change path. However on like the first, you can feel sleepy or over relaxed with the second"* for whom the second condition was SJ. This result could be related to our study setup, for which the goal configuration was constant throughout the experiment; in a real scenario, users would have chosen the goal themselves, either via waypoint navigation (see Section 1) or via choosing the destination on a minimap.



(a) (b)

Figure 14. Example paths taken using (a) SJ and (b) SW. Red parts are where the participant used their controller to give input.

Another reason why participants preferred SJ, related to the feeling of control, was the users ability to stop. Whereas with SJ the user had full control over the speed and orientation of the robot, thus allowing them to stop, with shared control, the robot always tried navigating towards the goal. The base implementation of the shared control allows for the speed to be controlled, we deliberately chose to keep the speed of the robot locked when using the shared control: we were worried about too much freedom of choice for the participants to confound the study with multiple simultaneous locomotion modalities. However, if they had had more control over the robot's destination and stopping, there is a chance users would have felt more in control even in the SJ and SG conditions.

We also deliberately chose not to include waypoint navigation. With it, the participants could have chosen wheter to use shared control or waypoint navigation depending on the size of the obstacle: however, giving them this freedom would have made analysis of the results more difficult. Several participants noticed that the proposed method was especially useful for small corrections *"It felt that it would be less laborious to just make small adjustments to the path when needed rather than drive manually."*, which is indeed the intended use case. Fig. 14 presents an example of this case such that SJ is used to make small alterations to the path whereas SW is used along larger portions of the path executed by the robot.

Even though several participants mentioned that their preference was affected by previous experience with games (*"It did not demand much mind effort to accomplish. and having joystick in the hand is a good experience from past playing video games."*), we did not find any correlation between gaming or VR background and preference of control method.

Since the main contribution of this paper is the underlying shared control mechanism, which can be used with either immersive or regular telepresence robots, the gesture condition, mainly meant for immersive telepresence robots, was left last as exploratory. Whereas using only gesture motions to control a robot might not be viable, as using extensive gesture motions for control is not encouraged in VR, using the shared control mechanism should not be a problem. With the ability to do small adjustments to the path without a need to take over all the motions of the robot, the user can control the robot's path with minimal effort even when using the gesture motions. Also, the joystick of the HMD controller has limited movement, and is quite small and not very accurate, which could be difficult to use for non-gamers or people

with medical conditions. Using your whole hand to give the input gives possibility to use much larger movements as inputs, increasing the accuracy and the ability to do fine motions with the robots even with limited control over your body. There are also interesting reasons in the qualitative data for preferring the gestures, such as *"It was more comfortable, a lot easier and it felt more real"*. The "felt more real" part could be related to *presence*, which is one of the main reasons for using HMD for telepresence. Thus, future research on whether body-based control increases the feeling of presence would be interesting.

6. CONCLUSION

This thesis presented a novel shared control method named HI-DWA. It extends the DWA approach for path tracking with the inclusion of a critic that takes into account the input given by a user. The method is aimed especially for immersive telepresence robots but it can be used in other kinds of robots too. The method allows the user to influence the trajectories the robot is taking and adjust its path to one's own liking. We performed a user study in VR, in which 32 participants compared the proposed method using joystick against switching between autonomous and manual modes. They used the different methods to avoid regions not visible to the robot's sensors. As an exploratory task, the participants also tested the proposed method using gesture motion controls. The results showed that the shared control was found to be easier, but the preference was split equally between the methods; the feeling of control was often mentioned as the reason for users who preferred the switching, even though there was no statistically significant difference in a Likert-scale question of feeling of control. The exploratory gesture motion controls was found to be harder to use and more physically demanding but still almost half of the participants preferred it over the shared control with joystick and the switching. The gesture motion controls were also described to feeling more real than the other methods, which shows that future research could be done on the gesture based controls and their effect on immersion and presence for immersive telepresence.

7. REFERENCES

[1] (2022), Double 3. URL: https://www.doublerobotics.com/.

[2] (2022), The future of remote collaboration. URL: https://gobe.blue-ocean-robotics.com/.

[3] Stoll B., Reig S., He L., Kaplan I., Jung M.F. & Fussell S.R. (2018) Wait, can you move the robot? examining telepresence robot use in collaborative teams. In: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, pp. 14–22.

[4] Kruse T., Pandey A.K., Alami R. & Kirsch A. (2013) Human-aware robot navigation: A survey. Robotics and Autonomous Systems 61, pp. 1726–1743.

[5] Bangay S. & Preston L. (1998) An investigation into factors influencing immersion in interactive virtual reality environments. In: Virtual environments in clinical psychology and neuroscience, IOS Press, pp. 43–51.

[6] Rae I. & Neustaedter C. (2017) Robotic telepresence at scale. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 313–324.

[7] Baker G., Bridgwater T., Bremner P. & Giuliani M. (2020) Towards an immersive user interface for waypoint navigation of a mobile robot. In: The Second International Workshop on Virtual, Augmented and Mixed Reality for Human-Robot Interaction.

[8] Fox D., Burgard W. & Thrun S. (1997) The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine 4, pp. 23–33.

[9] Kristoffersson A., Coradeschi S. & Loutfi A. (2013) A review of mobile robotic telepresence. Advances in Human-Computer Interaction 2013.

[10] Minsky M. (1980) Telepresence.

[11] Botev J. & Rodríguez Lera F.J. (2021) Immersive robotic telepresence for remote educational scenarios. Sustainability 13, p. 4717.

[12] Carranza K.A.R., Day N.J.B., Lin L.M.S., Ponce A.R., Reyes W.R.O., Abad A.C. & Baldovino R.G. (2018) Akibot: A telepresence robot for medical teleconsultation. In: 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), IEEE, pp. 1–4.

[13] Slater M. (2018) Immersion and the illusion of presence in virtual reality. British Journal of Psychology 109, pp. 431–433.

[14] LaViola Jr J.J. (2000) A discussion of cybersickness in virtual environments. ACM Sigchi Bulletin 32, pp. 47–56.

[15] LaValle S.M. (2021) Virtual reality. Cambridge University Press.

[16] LaViola J.J. (2000) A discussion of cybersickness in virtual environments. ACM SIGCHI Bulletin 32, pp. 47–56.

[17] Kemeny A., George P., Mérienne F. & Colombet F. (2017) New VR navigation techniques to reduce cybersickness. Electronic Imaging 2017, pp. 48–53.

[18] Al Zayer M., MacNeilage P. & Folmer E. (2018) Virtual locomotion: a survey. IEEE transactions on visualization and computer graphics 26, pp. 2315–2334.

[19] Mimnaugh K.J., Suomalainen M., Becerra I., Lozano E., Murrieta-Cid R. & LaValle S.M. (2021) Analysis of user preferences for robot motions in immersive telepresence. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 4252–4259.

[20] Sutherland I.E. (1968) A head-mounted three dimensional display. In: Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, AFIPS '68 (Fall, part I), ACM, New York, NY, USA, pp. 757–764. URL: http://doi.acm.org/10.1145/1476589.1476686.

[21] Zachara M. & Zagal J.P. (2009) Challenges for success in stereo gaming: a virtual boy case study. In: Proceedings of the international conference on Advances in Computer Enterntainment Technology, Acm, pp. 99–106.

[22] Anthes C., García-Hernández R.J., Wiedemann M. & Kranzlmüller D. (2016) State of the art of virtual reality technology. In: 2016 IEEE Aerospace Conference, pp. 1–19.

[23] Boletsis C. (2017) The new era of virtual reality locomotion: A systematic literature review of techniques and a proposed typology. Multimodal Technologies and Interaction 1, p. 24.

[24] Lavalle S.M. (2022) VIRTUAL REALITY. URL: http://vr.cs.uiuc.edu/.

[25] Kennedy R.S., Lane N.E., Berbaum K.S. & Lilienthal M.G. (1993) Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. The international journal of aviation psychology 3, pp. 203–220.

[26] Davis S., Nesbitt K. & Nalivaiko E. (2014) Systematic review of cybersickness. In: ACM International Conference Proceeding Series, ACM Press, New York, New York, USA, pp. 1–9. URL: http://dl.acm.org/citation.cfm?doid=2677758.2677780.

[27] Dragan A.D. & Srinivasa S.S. (2013) A policy-blending formalism for shared control. The International Journal of Robotics Research 32, pp. 790–805.

[28] Chiou M., Stolkin R., Bieksaite G., Hawes N., Shapiro K.L. & Harrison T.S. (2016) Experimental analysis of a variable autonomy framework for controlling a remotely operating mobile robot. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 3581–3588.

[29] Petousakis G., Chiou M., Nikolaou G. & Stolkin R. (2020) Human operator cognitive availability aware mixed-initiative control. In: 2020 IEEE International Conference on Human-Machine Systems (ICHMS), IEEE, pp. 1–4.

[30] Storms J.G. & Tilbury D.M. (2014) Blending of human and obstacle avoidance control for a high speed mobile robot. In: 2014 American Control Conference, pp. 3488–3493.

[31] Pappas P., Chiou M., Epsimos G.T., Nikolaou G. & Stolkin R. (2020) Vfh+ based shared control for remotely operated mobile robots. In: 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), IEEE, pp. 366–373.

[32] Chiou M., Talha M. & Stolkinl R. (2019) Learning effects in variable autonomy human-robot systems: how much training is enough? In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), IEEE, pp. 720–727.

[33] Wang H. & Liu X.P. (2014) Adaptive shared control for a novel mobile assistive robot. IEEE/ASME Transactions on Mechatronics 19, pp. 1725–1736.

[34] Krotkov E., Simmons R., Cozman F. & Koenig S. (1996) Safeguarded teleoperation for lunar rovers: From human factors to field trials. In: IEEE Planetary Rover Technology and Systems Workshop, vol. 26, vol. 26, p. 28.

[35] Luo J., Lin Z., Li Y. & Yang C. (2020) A teleoperation framework for mobile robots based on shared control. IEEE Robotics and Automation Letters 5, pp. 377–384.

[36] Jiang J. & Astolfi A. (2014) Shared-control for the kinematic model of a mobile robot. In: 53rd IEEE Conference on Decision and Control, IEEE, pp. 62–67.

[37] Fong T., Thorpe C. & Baur C. (2001) A safeguarded teleoperation controller. In: IEEE International Conference on Advanced Robotics (ICAR), CONF.

[38] Dijkstra E.W. et al. (1959) A note on two problems in connexion with graphs. Numerische mathematik 1, pp. 269–271.

[39] Hart P.E., Nilsson N.J. & Raphael B. (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4, pp. 100–107.

[40] LaValle S.M. et al. (1998) Rapidly-exploring random trees: A new tool for path planning .

[41] Macenski S., Dwb controller. URL: `https://github.com/ros-planning/navigation2/tree/main/nav2_dwb_controller`.

[42] Lu D., dwb local planner. URL: `https://github.com/ros-planning/navigation2/tree/main/nav2_dwb_controller`.

[43] Borenstein J. & Koren Y. (1989) Real-time obstacle avoidance for fast mobile robots. IEEE Transactions on systems, Man, and Cybernetics 19, pp. 1179–1187.

[44] Gerkey B.P. & Konolige K. (2008) Planning and control in unstructured terrain. In: ICRA workshop on path planning on costmaps, Citeseer.

[45] Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R., Ng A.Y. et al. (2009) Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol. 3, Kobe, Japan, vol. 3, p. 5.

[46] Macenski S., Martin F., White R. & Ginés Clavero J. (2020) The marathon 2: A navigation system. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[47] Macenski S., Martín F., White R. & Clavero J.G. (2020) The marathon 2: A navigation system. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2718–2725.

[48] (2020), Navigation 2. URL: `https://navigation.ros.org/`.

[49] Ros-tcp-connector. URL: `https://github.com/Unity-Technologies/ROS-TCP-Connector`.

[50] Patton M.Q. (2005) Qualitative research. Encyclopedia of statistics in behavioral science .