FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Juho Kurula
Jeremias Körkkö
Kalle Veijalainen

COLLABORATIVE VIRTUAL ENVIRONMENT FOR HUMAN-ROBOT INTERACTION AND NAVIGATION

Bachelor s Thesis
Degree Programme in Computer Science and Engineering
May 2022

Kurula J., Körkkö J., Veijalainen K. (2022) Collaborative Virtual Environment for Human-Robot Interaction and Navigation. University of Oulu, Degree Programme in Computer Science and Engineering, 35 p.

ABSTRACT

In this work, we present a data-gathering tool for virtual reality human-robot interaction focusing on the trajectories of the participants. Unity, Robot Operating System, Photon PUN, and Oculus were utilized to create a lightweight multiplayer environment for various studies. The system supports various amounts of humans and autonomous as well as teleoperated robots. The data from these interactions can be collected and further analysed to find possible differences in human behaviour. Positional and orientational data proved to be accurate. Measured latency of 200ms was found to be sufficient for the trajectory collection.

Human-robot interaction studies are often restricted because of the challenges regarding large datasets, time, and financial matters. With the implementation of virtual reality, many of these challenges can be addressed. Virtual reality offers safe and easier ways to research situations that could be dangerous or impossible to produce in real life.

Keywords: virtual reality, VR, HRI, human-robot interaction, VR headset, data-gathering, teleoperation, autonomous, robotics

Kurula J., Körkkö J., Veijalainen K. (2022) Interaktiivinen virtuaaliympäristö ihmisten ja robottien välisen vuorovaikutuksen tutkimiseen. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 35 s.

TIIVISTELMÄ

Tässä työssä esittelemme tiedonkeruutyökalun ihmisten ja robottien välisen vuorovaikutuksen tutkimiseen virtuaalisessa todellisuudessa. Työkalu on kevyt moninpeli alusta, joka hyödyntää Unity-, Robot Operating System-, Photon PUN-, sekä Oculus-teknologioita. Järjestelmä tukee yhtä aikaisesti useita osallistujia, ihmisiä sekä teleoperoituja tai autonomisia robotteja. Ihmisten ja robottien välisistä kanssakäymisistä saatua dataa voidaan analysoida ja pyrkiä selvittämään mahdollisia muutoksia ihmisen käyttäytymisessä erilaisissa tilanteissa. Sijainti- ja orientaatiodata osoittautui tarkaksi. Mitattu 200ms viive on riittävä liikeratojen seuraamiseen.

Ihmisten ja robottien vuorovaikutusta tutkivia tutkimuksia kuitenkin usein rajoittaa useat tekijät, kuten suuret tietoaineistot, pitkät tutkimusten kestot ja suuret kulut. Virtuaalisen todellisuuden hyödyntäminen tutkimuksissa voi auttaa näiden ongelmien ratkaisemisessa. Se tarjoaa myös turvallisempia ja helpompia vaihtoehtoja mahdollisesti vaarallisten tai muuten mahdottomien oikean elämän tilanteiden toistamiseen.

Avainsanat: virtuaalitodellisuus, VR, ihmisten ja robottien välinen vuorovaikutus, tiedonkeruu, autonominen, robotiikka

TABLE OF CONTENTS

FOREWORD

We would like to thank Başak Sakçak and Markku Suomalainen for being active and helpful supervisors at all stages of this project. We would also like to thank Timo Ojala for coordinating this course, and also for his feedback and helpful tips during the meetings.

Oulu, May 4th, 2022

Juho Kurula
Jeremias Körkkö
Kalle Veijalainen

LIST OF ABBREVIATIONS AND SYMBOLS

6DOF	Six degrees of freedom
API	Application programming interface
CAD	Computer aided design
DDS	Data distribution service
DLL	Dynamic link library
HMD	Head mounted display
HRI	Human robot interaction
ORCA	Optimal reciprocal collision avoidance
PC	Personal computer
PUN	Photon Unity networking
ROS	Robot operating system
RoSAS	Robotic social attributes scale
SM	Social momentum
SUS	Slater-Usoh-Steed
TCP	Transmission control protocol
TE	Teleoperation
UBICOMP	Center for Ubiquitous Computing at University of Oulu
USB	Universal serial bus
VR	Virtual reality

1. INTRODUCTION

Human-Robot Interaction (HRI) is often mistakenly seen as a new and emerging field, but the concept has been around as long as there has been the concept of robots themselves. The man who came up with the term robotics in the 1940s, Isaac Asimov, was already asking questions such as "What kind of relationship can a person have with a robot?" which is a question at the very core of all HRI research to date [1]. Although these questions were more science fiction than reality back then, it indicates that there already was a difference between robotics and HRI.

Even though HRI as a field is not new and emerging, it is evolving rapidly. Hundreds of papers are published each year by thousands of contributors and by many different professional societies. Within a span of 80 years, applications have evolved from master-slave remote item handling devices to gesture-operated mobile robots performing an interactive clean-up task in changing light conditions [2]. First robot applications were actually designed to handle relatively simple tasks such as handling radioactive objects to avoid exposing humans to radiation. Today, scientists are looking to solve global issues like aging societies with the usage of robots, for example, elderly and health care. Such tasks are difficult to be automated perfectly, however, they could be executed in cooperation with humans and robots.

There are multiple challenges regarding traditional HRI research. For example, the collection of the datasets for later use in machine learning has proven to be a significant limitation itself which can take up to two months [3]. Other challenges regarding HRI research are the significant amount of time to practice the subject experiment and the substantial consideration of an experimental design itself [4]. The field has now started to seek solutions from Virtual Reality (VR) applications and simulations to reduce the cost of the projects. One definition for VR could be: Inducing targeted behavior in an organism by using artificial sensory stimulation, while the organism has little or no awareness of the interference [5]. And in terms of this project, the immersion is the main component of that definition. Immersion is achieved with a virtual reality headset. Headsets replace all of the real world in the field of view with visuals from virtual reality. VR provides the effects of a concrete existence without actually having a concrete existence. On a computer, virtual reality is primarily experienced through two of the five senses: sight and sound.

The introduction of VR to HRI research has many advantages. Firstly, it can help to reduce the cost of the research, as experimental facilities are not required in a real environment. Secondly, the visualization of arbitrary information and situations that are not possible in reality or are otherwise hazardous or dangerous to reproduce, such as the playback of past experiences can be achieved, and thirdly, it provides access to multiple immersive and natural interfaces for robot/avatar teleoperation systems [4]. The virtual reality system presented in this work provides, for example, a solution to study the interactions between humans and multiple robots without the need to invest in multiple actual robots.

The motivation of this project is to create a virtual multiplayer environment to test the interactions between humans and mobile robots in shared environments. The system is required to utilize Robot Operating System 2 (ROS) and Unity to provide an environment for multiple humans and robots to interact. It is required to support teleoperated and autonomous robots that are connected to Unity via ROS as well as

users with Oculus VR headsets. Humans and robots need to be able to move around in the environment freely. The system is also required to be capable of collecting data of the human and robot trajectories. The data collected from these interactions can be used in further research regarding the possible differences in human behaviour when interacting with teleoperated and autonomous robots; for instance, proxemics (the study of spatial distances between individuals), activity, speed, and movement patterns.

In this project report, we rst discuss related work and technologies in Chapter 2. In Chapter 3 and Chapter 4 we explain the design and implementation respectively. Evaluation and discussion are in Chapter 5, and the conclusion of the project will be in Chapter 7.

2. RELATED WORK

This section presents a brief overview of previous HRI studies. We review the research environments and technologies related to the system created in this work. This section also introduces technologies required in the implementation.

2.1. VR in HRI Research

Virtual reality has been utilized in various HRI studies. It allows easier data collection and access to environments that can be hard to access otherwise. Li et al. [6] as well as Yang et al. [7] studied proxemics and social distances in virtual environments. Both of these studies also included a robot approaching a human or a group of humans. Besides human-robot interaction, VR is used in other sub elds of robotics for data gathering. In the research done by Koller et al. [8] data was gathered from a human movement in a virtual reality kitchen to then train the robot later on. Even though the usage of virtual reality can bring some real bene ts to the studies, some researchers still prefer doing things the old way. For example, Mavrogiannis et al. [9] focused on how humans would react to different behaviour patterns of a robot and what effect that would have on overall performance for both humans and the robot, in a traditional lab environment. Had this study been executed in a virtual reality environment, instead of post-it notes and markers, the environment could have been more immersive and the tasks more authentic.

The research was limited to the complexity of the real-world pedestrian environment because the study was done in lab conditions. Different to our study, this research did not include virtual reality. The participants had to imagine being in a factory even though they were inside a classroom. An easel was representing a machine with sticky notes and a marker pen represented the maintenance tasks. The measurements were done both objectively and subjectively. Objective measures are in this case all the tracked human and robot trajectories recorded through an overhead motion capture system. Subjective measures are questionnaires about ratings of participants, and impressions of the robot s intelligence, safety, and personality. Mavrogiannis et al. [9] found evidence that human acceleration is lower and their movement is not as irregular when they are navigating around an autonomous robot, compared to a teleoperated one. They also found out, against their predictions, that there was a lack of support to con rm that the teleoperation strategy on the robot would be humans preferred choice. The movements of the participants and the robot were captured with an overhead motion capture system and a videotape if the participants had given consent. Six high-accuracy and high- delity cameras tracked the re ective markers attached to the construction helmets worn by the participants. With the implementation of virtual reality, there would have not been a need to imagine the factory as a realistic virtual reality factory could have been created. Also, the data collection could have been done with the data from the VR headset, and consent of a participant would not have been needed, since a video of a real human being would not be needed. Possible dangerous real-world situations can be avoided as well.

Li et al. [6] discovered that people tend to prefer a longer distance and personal space between the robot in virtual reality than in the real world. The difference in proxemics

could be caused by the ego-centric distances that are perceived as compressed in virtual reality [10, 11]. Participants also perceived the virtual robot to be remarkably more discomforting and have a lower feeling of presence in virtual reality. The reason for this might be the quality of the 3D model robot was not good enough or as good as in the real world. The results also showed that the addition of spatial sound did not increase the sense of presence in VR and the reason for that could be the social presence of the researcher who gave instructions now and then to participants outside of virtual reality.

The research done by Li et al. [6] had three different manipulations: the presentation methods meaning of comparing real-world and virtual reality proxemics, the visual familiarity of the physical environment in VR, and spatial sound. Their nal sample included 60 participants and they were exposed to two trials, one real-world trial, and one VR trial. In VR trials they were altered to explore the in uence of visual familiarity and spatial sound. In the experiment, participants were holding an HTC Vive controller to enable the robot to move forward toward the user when pressing the trigger button. After the users minimum comfort distance was exceeded, meaning that the user was feeling uncomfortable, the trigger button would be released and robot s approach would stop and the proxemics could be measured. HTC Vive controllers and HTC Vive Tracker were also used to gather the data from the proxemics of HRI. HTC Vive Tracker was placed in the back of the robot and the controller was in the user s hand. Unlike our study, the environment is not as immersive because there is only one robot and one human participant who could only see the robot. There was also no movement included in the environment except only for the robot which moved towards the user when pressing the button indicating teleoperation. Compared to our work, there are no tests with autonomous robots which could have made a difference in this study affecting the proxemics, because a robot moving autonomously towards the user might create more uncomfortable feelings.

Similar to the study done by Mavrogiannis et al. [9] the study done by Li et al. [6] also included subjective measures such as the Robotics Social Attributes Scale (RoSAS) and Slater-Usoh-Steed (SUS) questionnaire [12, 13]. RoSAS was used to evaluate the perception of the robot. It is an empirically validated method of measuring the perception of the robot. In this study, they used two out of three factors that are competence and discomfort because the warmth factor is not related to the task. The other questionnaire SUS comprises six questions that were used to measure the feeling of presence in this study.

Koller et al. [8] created an open-source virtual reality simulator called Virtual Annotated Cooking Environment (VACE) for object interaction tasks in a richly furnished, interactable virtual kitchen environment. This project was done using Unity. The VR simulator creates thoroughly annotated video sequences of a virtual human avatar. These human avatars are controlled by real humans with a HTC VIVE headset, chest tracker, and controllers. The body of the avatar is animated through an inverse kinematic system to generate realistic arm, leg, and torso behavior. With this hardware, users movement is tracked and the trajectory data is made into datasets. In addition to real live recordings, Koller et al. [8] say datasets generated in virtual environments are valuable for activity and plan recognition, reinforcement learning, learning from demonstration, and semantic segmentation research. With this simulator and data gathered from the simulation, machine learning can be utilized and agents can be made

to create future service robots in human homes which can perform daily tasks in the kitchen and cooperate with their human partners.

Similar to our study, this research uses Unity to create the virtual environment. In addition, this project collects data from the VR hardware for future HRI studies. Different to our study, this project does not include teleoperated or autonomous robots interacting with humans or any kind of human-robot interaction in the virtual environment. Similar to our project, Yang et al. [7] used a virtual reality environment and virtual reality headset to gather the data which people in real life were using. They used humanoid characters to represent humans in a virtual world. In the research procedure, a group was formed, which was then approached by a teleoperated character. From these interactions, they attempted to gure out which was the most socially acceptable manner to make individuals in the group feel comfortable using attention- and graph-based neural networks from the gathered trajectory data.

2.2. Research Environments

Various types of virtual research platforms have been utilized in the past, from ones built on game engines to realistic VR models of real-world environments. For example, the Malmo Project uses Minecraft world to generate complex and intuitive 3D worlds for Arti cial Intelligence (AI) and robotics research [14]. These kinds of worlds focus more on the development and the studying of the agents. Then again, Project Malmo represents a more unusual approach to exposing agents to a range of environments. Worlds built-in virtual reality offers many possibilities for gathering data from versatile and arbitrary situations and environments. They especially offer new ways to observe realistic interactions between humans and robots. SIGVERSE is an example of this kind of platform that specializes in data collection from various interactions [4].

SIGVERSE shares a lot of the same technologies with our platform. Both use Unity as a game engine, with Photon PUN networking to provide multiplayer functionalities. Both also use ROS , introduced in chapter 2.3, as a bridge between the robots and Unity. In our platform ROS2 is implemented rather than the older ROS 1 in SIGVERSE. The main differences between these two versions are that ROS uses Python 2 whereas ROS2 uses Python 3, ROS only supports Ubuntu whereas ROS2 is supposed to support Windows 10 as well, and in ROS2 other than CMake projects can also be used. These changes make ROS2 a more compatible and sustainable choice[15]. The main difference between our platform and SIGVERSE is that SIGVERSE supports a wider range of actions and collects a multitude of data from them such as motion gestures, pictures, spatial information, and audio. Our platform is more lightweight and focuses solely on collecting spatial data of objects [4].

Another interesting environment was used by Li et al. [6] in a study focused on proxemics between virtual reality and the real world. Although Unity was also used, the system in question has some major differences. Unlike previously mentioned systems this one does not support multiplayer activities at all. There is only one robot and one human in the scene at once. The robot does not have the possibility to move freely or autonomously as it is controlled by the participant in the virtual environment who can move the robot only towards themselves by pulling a trigger on their VR

controller. Even though the setting is quite different, the purpose is quite similar; to collect proximity data of robots and humans. In addition, our system also collects the direction and speed of the objects. Our platform and SIGVERSE have support only for Oculus VR headsets, while the system in this study supports only HTC Vive VR headsets. This allows the system to sync the location of the robot with the HTC Vive tracker, that was placed on the physical robot for more accurate results.

2.3. ROS (Robot Operating System)

ROS is a free, open-source collection of hardware drivers, algorithms, networking models, and other tools used in robotics. The core idea of ROS is to make the existing software technology available and easily accessible for robotics study and development. Even though the name suggests so, ROS is not a traditional operating system. It is a middleware that runs inside a host operating system offering functions that are suchlike a traditional operating system provides [15, 16].

This work implements ROS 2. Original ROS 1 system creates a network maintained by ROS master, that keeps track of all activities in the network. Newer ROS 2 uses Data Distribution Service (DDS) technology [17]. Data Distribution Service integrates nodes in the network into a single system, lowering latency and enhancing reliability and scalability. Essentially it is a middleware protocol and API-standard [18]. ROS 2 DDS network is formed from several ROS 2 nodes processing data jointly and concurrently and it does not include a master [15, 17].

A node in a ROS 2 network is a piece of software that has its own modular function to take care of. This function can be rotating a wheel or sensing temperature through a sensor, for example. Data is transmitted between the nodes in the network using topics, services, actions, and parameters. These communication frameworks offer a versatile playbook of techniques for sharing data ef ciently within the network. Event-triggered data sharing (topics), request-based call-and-response communication (services), and continuous communication through actions, which are constructed by combining topic and service communication logic. Parameters are a set of unique settings for each node and are used to con gure the network [15].

2.4. Unity

Unity is a 3D real-time simulation platform that consists of multiple renderers, physics engines, and Unity Editor which is a graphical user interface. It was developed by Unity Technologies in 2005 and is commonly known engine for the development and creation of games [19]. Besides games, Unity can also be used to create virtual reality applications. A project in Unity consists of multiple windows including scene, game, hierarchy, project, and inspector. In scenes, you can structure and debug the project in a logical and modular manner. In the hierarchy and scenes, you can add various types of game objects, which themselves contain different types of components. Components included in a game object will determine the object s function and behaviour. For example, components can be mesh lters, renderers, colliders, or joints. Using components will make the game object either static or dynamic. The designer can

change the scene window to see the progress of the project from different perspectives and see how different game objects behave [20].

Unity uses an object-oriented framework for runtime and design-time scripting in three languages: Boo (By Unity), JavaScript, and C#. In addition to those three programming languages, other .Net languages can be used with Unity under certain prerequisites [20]. Runtime and Editor are the two types of object classes used in Unity. Runtime classes contain user-written scripts that are used in-game objects. One or multiple of the scripts can be attached to one or multiple game objects and change the behaviour of the object. Users can use scripts in Editor classes to add menu items to the default Unity menu system [21].

Any item that can be used within a project, no matter if it is a scene, a game object, or a component, can be turned into an asset. An asset is a representation of that item. It can be either imported from outside of Unity or made inside of Unity. Files from common Computer-Aided Design (CAD) software can be imported into Unity and be made an asset. Assets can also be purchased or downloaded for free from Unity Asset Store. Assets can be reused within one project and be copied into other projects [20].

2.5. Oculus

Oculus VR Inc. was founded in 2012 by Palmer Luckey, Brendan Iribe, Michael Antonov, and Nate Mitchell to develop a VR headset for video gaming known as Oculus Rift. In 2014, Facebook Inc., nowadays known as Meta Platforms, bought Oculus for 2,3 billion dollars in cash and stock [22]. Oculus produces virtual reality headsets including head-mounted displays (HMD) and Oculus Touch controllers. Its bestselling products include the Oculus Rift and Oculus Quest lines [23].

The Oculus Quest comes up with two different models: Oculus Quest and Oculus Quest 2. They can run games and software also wirelessly because they have integrated computers inside the HMD. The HMDs are powered by Qualcomm Snapdragon mobile processor platform and the operating system is Oculus Quest system software, based on Android source code [23].

The Oculus Quest supports both orientation and position tracking with two six degrees of freedom (6DOF) controllers and a headset, using internal sensors and an array of cameras in the front of the headset rather than external sensors. Therefore, the pose (of a user s head and hands) in 3D can be tracked, that is the coordinates along the x (horizontal), y (vertical), and z (depth) axes, which determine the position, in addition to the pitch, yaw, and roll, which determine the orientation. The 6DOF movement allow user to integrate virtual hands to interact with VR environments [24, 23]. The Quest lines are also capable of running with Oculus-compatible VR software running on a desktop computer when connected over USB [23].

3. DESIGN

The purpose of the project is to create a comparably lightweight data gathering tool for HRI research focusing on the trajectories of humans and robots within interaction situations. The system needs to ful l the given requirements to gather data in a wanted format in a controlled setting. Multiplayer functionalities must be supported. Multiple participants need to be able to join a room and interact with each other as players playing the game. Rooms are separate instances of a certain level or map inside a game or application within a server. Multiple instances of a single map can exist simultaneously and each of these instances is a separate room. A participant joining the room has to have a choice to join in as a human or as a robot. Human inputs are given through Oculus headset and handheld controllers. Robot inputs are sent as Transmission Control Protocol (TCP) messages from ROS. The controller giving the instructions through ROS can be a participant using a keyboard (teleoperation mode) or an autonomous script (autonomous mode). All useful data from the interactions must be recorded. In our case, this corresponds to recording the positions and orientations of the game objects and the directions that they are looking at, at discrete time instances.

The requirements set for this project set some constraints for the design and architecture of this project. This chapter gives insight into the architectural and design features implemented that are a direct or indirect result of the requirements and speci cations given.

3.1. Use Case

This platform is created to observe the behaviour of participants in a controlled environment. Multitude variants of experiments can be created to research the effects of changing certain variables and/or variable combinations on the behaviour of the subject. Creating a virtual reality tool for this purpose cuts costs and saves time for the researchers.

By changing the method the robots are controlled (autonomous/teleoperated), the interaction behaviour can be analysed from the collected data. Human avatars are always controlled by a human so they can be used as a reference point. There is no way to tell in-game if a robot is autonomous or teleoperated. This uncertainty can be leveraged and used to observe if there are any differences or patterns in the interactions of the objects who are the same or different in-game. Also, the setting can be switched on its head by telling or lying to the subject about the status of the robot(s) and seeing if it has any effects. These aspects can be replicated with many different combinations or number of subjects, characters, and settings which makes this platform useful for many purposes. Other features can also be changed within the scene editor without breaking the game, for example, the map or relational sizes of the characters can be changed if needed to test an even larger amount of variables and different parameters. Possible use cases are discussed more speci cally in Future Work in Chapter 6.2.

3.2. Models

There are two types of character models in our environment, the robot model in Figure 1 and the human prefabs in Figure 3. The model of the robot is made by the UBICOMP team at the University of Oulu and the human prefabs, made by IRONHEAD Games, were acquired while attending a VR multiplayer course[1]. The robot is a differential drive system with the driving wheels hidden under the chassis. The wheels are controlled with an AGVController.cs script which calculates the torque input for each wheel to track the control commands (linear and angular velocities of the robot). The wheels then move the robot using game engine physics which take into account aspects such as friction. This script allows the robot to move forwards and backward, turn left and right, and adjust the speed of the robot. The robot also has a LaserScanSensor.cs script which is used for navigation and collision avoidance of the autonomous robot. Scanning is visualised by the pink markers on the ground, seen in Figure 2.

Figure 1. Robot prefab asset

The robot is an accurate replica of the physical robot at the UBICOMP of ce. The only differences are that in the Unity version, the script AGVController.cs is simulating the actuator that drives the wheels, and the model of the robot was missing the rod which is holding the camera, but we added it. The camera of the robot is placed approximately at the eye level of an average person to make the interactions feel more real. Teleoperated and autonomous robots are identical, Oculus users or even Unity cannot distinguish the difference between teleoperated and autonomous robots based on their inputs or appearances. Only the movement patterns may differ.

To easily separate robots from human participants human avatars needed to be added. For this project, we attended a VR multiplayer course, which gave us access to IRONHEAD Games multiplayer VR assets. These assets included human prefabs in Figure 3. Although they are very simplistic they have many good attributes which made us choose them. Prefabs do not have arms or legs. They have the same body parts as Oculus VR headset tracks: position and orientation for both hands from controllers

[1]Multiplayer Virtual Reality (VR) Development With Unity by Tev k Ufuk DEMIRBAŞ and IRONHEAD Games, URL: https://www.udemy.com/course/multiplayer-virtual-reality-vr-development-with-unity/

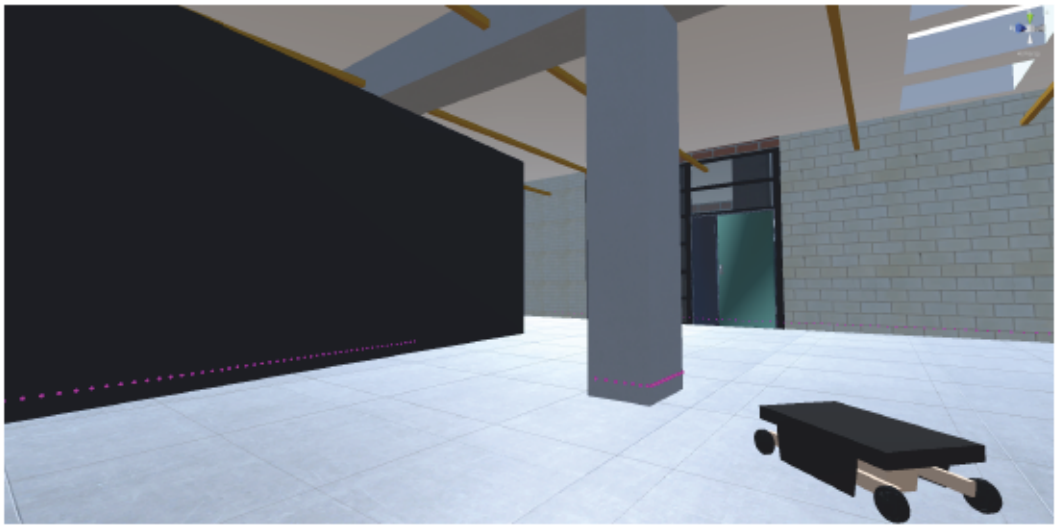

Figure 2. Robots scanning markers.

Figure 3. IRONHEAD Games human avatar prefabs in idle poses.

and the same information for the head from HMD. Head rotation is applied to the body as there is no separate sensor to read body orientation. Hands can move freely. This allows uent movements and easy gesture expressions. We believe this encourages participants to act more naturally around each other as there are no stiff statue-like characters or unnatural clunky animations.

3.3. Versatile Environment

One of the major bene ts of the system is its modularity. Different modules and functionalities of the system are kept apart from each other. This means that modifying or entirely altering one functionality does not always affect or break the other functionalities of the system. This allows easy access to diverse environments and research settings.

The whole building and all of its components can be found under one single GameObject. It does not contain any scripts or other connected functionalities, only

mesh to avoid objects moving through the structures. This means that changing the room or the building in practice can be done by simply deleting the old object and importing a new one to the project. Changing the avatar requires a bit more work compared to the room. It requires the corresponding GameObjects such as the body, hands, and the head to be reconnected to the scripts that handle the movement and synchronization of the human.

Figure 4. Tellus environment used in development phase.

In the development phase of the project, we used a model of the Tellus, found in the Linnanmaa campus of the University of Oulu. It is a precise 3D model of the Tellus, made with Blender, by the UBICOMP team. A picture of the environment can be seen in Figure 4.

3.4. Risks

Risk	Description	Likelihood	Impact	Preventive actions	Corrective
Schedule	Project does not stay on schedule	unlikely	major	Proactive scheduling and project management, avoid procrastination	Try to pick up the pace
Unity	Third party software support come to an end	extremely rare	catastrophic	-	Switch to some other game engine
Photon PUN	Third party software support come to an end	rare	major	-	Switch to some other Unity extension
Oculus	Third party software support come to an end	extremely rare	catastrophic	-	Switch to some other VR headset manufacturer
ROS	Third party software support come to an end	rare	moderate	-	As ROS is an Open-source software, development can still continue without active support

Figure 5. Risk assessment.

There are a few factors that could possibly threaten the nalization of our project. Firstly, a factor that is involved with every project, is the schedule. As we don t really have previous experience with the involved technologies, there is a reasonable probability that we encounter some issues that take a lot of time to overcome and it could possibly put the schedule at risk. Then there are risks involved every time a third-party software is involved. For our project, it means that we are more or less dependent on Unity, Photon PUN, Oculus, and ROS. Risks and their attributes can be found in the Figure 5. The risk likelihoods and impacts are assessed based on if events occur during the project development phase.

4. IMPLEMENTATION

Even though the design is quite predetermined in terms of the direct and indirect requirements of the system, they do not cover all the functionalities. For example, the implementation of multiplayer functionality and data collection are not speci ed by the requirements or the subject. The con guration regarding whether to use a virtual machine or two computers is also just a preference. In this section, we explain the choices we made regarding those functionalities and the implementation of them.

4.1. Multiplayer

Multiplayer functionality is implemented in this project to allow multiple players to interact and collaborate in the same virtual space simultaneously. This is done using Photon PUN. PUN (Photon Unity Networking) is an extension asset for Unity created by Photon. It offers cross-platform backend functionalities for room-based Unity multiplayer games. It utilizes client to server architecture Figure 6. Users can connect to Photon Cloud through applications and games to interact with each other in real-time. Photon Cloud is a globally compartmentalized collection of servers running Photon Server -application maintained by Exit Games. As a service Photon Cloud takes care of server operations, hosting, and scaling so the developers can concentrate on their application [25].

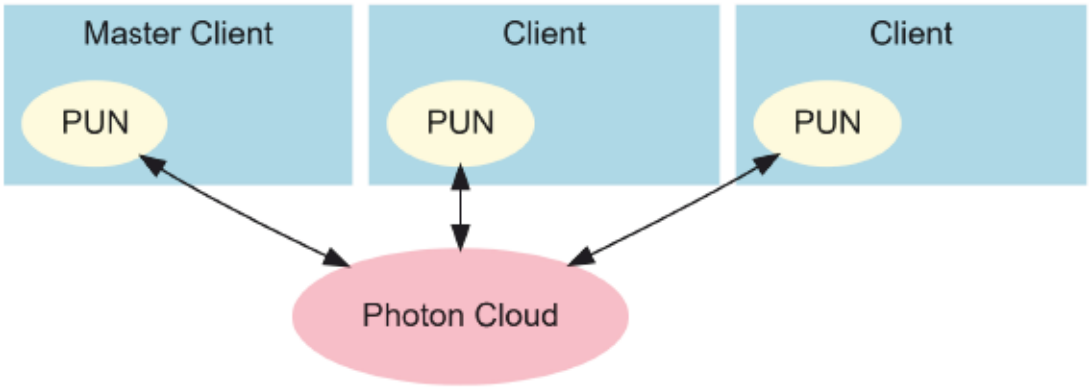

Figure 6. Communication between clients and Photon Cloud.

PUN package forms from API of three layers. The rst level contains the code, handling of network objects and method calls with remote clients, etc. The second layer handles server level: connections, matchmaking, callbacks, and so on. The lowest level implements dynamic link libraries (DLLs) [20]. DLLs contain program code and information that are meant to be used by multiple programs concurrently, for example, serialization and deserialization logic and protocols [25, 26].

We chose PUN because of the high level of service it offers. It extends the built-in networking of Unity and therefore, the APIs are very similar. Photon PUN documentation [25] includes a lot of pre-built methods and information about their functionality to make developing a multiplayer game easier and faster. A

lot of documentation and instructional material are available online. Also, the course we bought, Multiplayer Virtual Reality (VR) Development With Unity, utilized Photon PUN services. This allowed us to save time by utilizing the script MultiplayerSynchronization.cs and the prebuilt characters that were presented in Chapter 3.

The game reads inputs, either from the VR headset and controllers or commands from ROS (in case the character is a robot). According to the input, the game locally updates the rotation and position of the character. PUN collects this information and sends it from all participants to all participants. This information is used to move and rotate the corresponding characters equivalently on all players screens in the room. Actions are synced only between players within the same room. This creates the illusion of shared space.

4.2. Input Methods

When connected to a room as a human avatar, users have two ways of moving around. The primary method of control is just to move around physically in the real world with the headset. The movements are then reproduced in the virtual environment by the human avatar. As a secondary method of control, users can walk around as an avatar by using the Oculus Touch controller joystick. This method is not ideal in terms of research analysis as it is not as natural way of moving for human beings, and it can cause motion sickness, but it can be useful for example in certain situations where there are external constraints such as insuf cient amount of space to move around in.

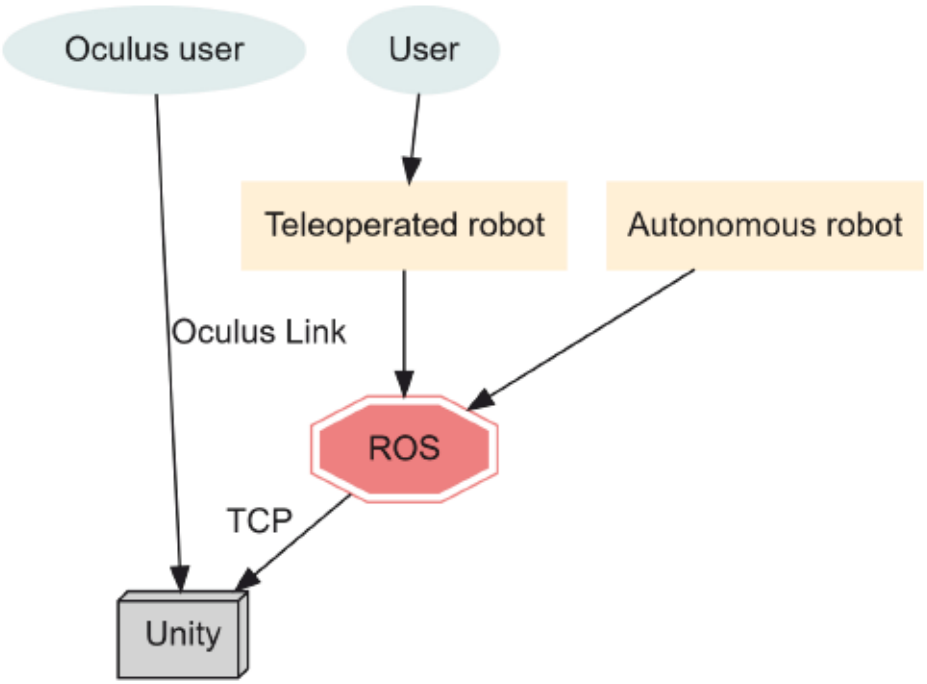

Figure 7. Three options for movement inputs.

As for the robots, there are also two ways to move around, which both appear identical from Unity s point of view: teleoperation and autonomous navigation. The

inputs that ROS sends can be from an autonomous navigation system or from the user s keystrokes and they are indistinguishable for Unity. Teleoperated robots are controlled with a keyboard and autonomous robots navigate using a predetermined map or separate runtime waypoints along with collision avoidance. Both control methods also support looking around freely with an Oculus VR headset which simulates a 360° camera.

Figure 8. Screenshot of the main menu

When a user starts the program, they are rst brought to the main menu, Figure 8. In the menu, the user has the option to choose whether to join the session as a human or a robot avatar and whether to collect the data locally or not. The main menu is only visible on the computer monitor, not on the VR headset and it is operated with a mouse. Users can click the data collection button to set it ON or OFF and then click the HUMAN or ROBOT button to spawn in as one. It is necessary to ask the user s choice for these options and the main menu is a simple and effective way to do so.

4.3. Data Collection

As stated before, the data collected by the system consists of position and the orientation of the players. More precisely, the system tracks the coordinates along the x and z axes, and the rotation about the y-axis for human avatars. Different to many other systems, the y-axis is the vertical axis in Unity Figure 9. If necessary, missing data elds can easily be collected as well, but to keep the system as lightweight as possible and to avoid collecting irrelevant data, those elds are not included originally. For example, the y-coordinate is excluded from the data because, in the current map, all movement happens at the same level vertically. A data sample can be seen in the Figure 10. There it can be seen that the system collects data from six data elds.

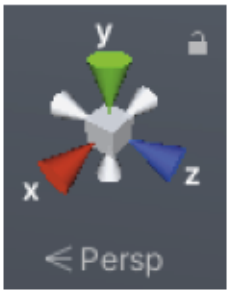

Figure 9. Coordinates in Unity.

	A	B	C	D	E	F
1	Column1	Column2	Column3	Column4	Column5	Column6
2	Time	ID	posX	posZ	rotY	robotCamY
3	3628402,509	player_3_human	-10	-5	318,1	-
4	3628402,509	player_2_robot	-7,9	-1,8	26,1	32,55253
5	3628402,509	player_1_robot	-11,8	-4,5	7,1	-
6	3628403,008	player_3_human	-10	-5	311,9	-
7	3628403,008	player_2_robot	-7,7	-1,3	26,1	32,54705
8	3628403,008	player_1_robot	-11,7	-4,3	8,5	-
9	3628403,51	player_3_human	-10	-5	307,9	-
10	3628403,51	player_2_robot	-7,5	-0,9	26	32,53302
11	3628403,51	player_1_robot	-11,7	-4,2	24,3	-
12	3628403,998	player_3_human	-10	-5	278,4	-
13	3628403,998	player_2_robot	-7,4	-0,5	354,6	0,577631
14	3628403,998	player_1_robot	-11,6	-4	31,9	-
15	3628404,499	player_3_human	-10	-5,1	265,3	-
16	3628404,499	player_2_robot	-7,4	-0,3	293,6	295,8494
17	3628404,499	player_1_robot	-11,4	-3,8	35,3	-
18	3628405,001	player_3_human	-10	-5	324,6	-
19	3628405,001	player_2_robot	-7,4	-0,3	265,9	263,817
20	3628405,001	player_1_robot	-11,3	-3,6	37,5	-
21	3628405,503	player_3_human	-9,9	-5	11,6	-
22	3628405,503	player_2_robot	7,4	0,3	255,2	230,7578
23	3628405,503	player_1_robot	-11,3	-3,6	61	-

Figure 10. Data sample.

Metadata consists of the two first columns. The timestamp is Photon network time which is synchronized with the server. The time can vary between servers, but inside a room, all clients have the same value. The timer starts from a random positive value between 0 and 4294967.295 seconds and wraps around when the maximum value is reached. The maximum value equals almost 50 days so the likelihood of duplicate timestamps is negligible. The system collects data once every half a second by default. If needed, this value can be changed from the TrajectoryCollector.cs script. The ID field consists of three components separated by underscores: the first part is a static text "player", the second part is the ActorID of the Photon session, and the third part is a robot or human keyword accordingly. The ActorID provides a unique identifier for each player that has connected to the session. Note, if a player who last joined disconnects, then joins again, their ActorID is incremented by 1. The posX and posZ data fields contain values of the x and z coordinates for both human and robot game objects. The rotY value represents the direction of the body. Human orientation is

tracked from the VR headset so it is also the direction the object is watching. The robot's 360° camera can watch in a different direction than the body is heading and therefore robotCamY value is collected separately to save the direction the robot user is looking for the local robot. Robot camera orientation is not synced over the network as it does not affect the appearance of the robot. This is why robot camera rotation can only be saved from the local object. The script iterates through all the players and writes their data at the moment of the execution.

When the data collection is enabled, the path where the trajectory_data.ssv file is located is printed to the debug log. It is in the persistent path of Unity, which usually is `C:\Users\<userprofile>\AppData\LocalLow\<companyname>\<productname>`. In the development stage, the companyname was DefaultCompany and productname ACP1.

4.4. Unity-ROS Connection

In the development stage of this system, a virtual machine was utilized to run the Linux distribution. Practically speaking, this means that ROS was running on the virtual machine Linux, and the Unity project was running on native Windows 10. As for the Linux distribution, Ubuntu was used while developing the system. This way the communication between the two operating systems was done with TCP (transmission control protocol) connection through the loopback connection. Figure 11 visualizes this connection.

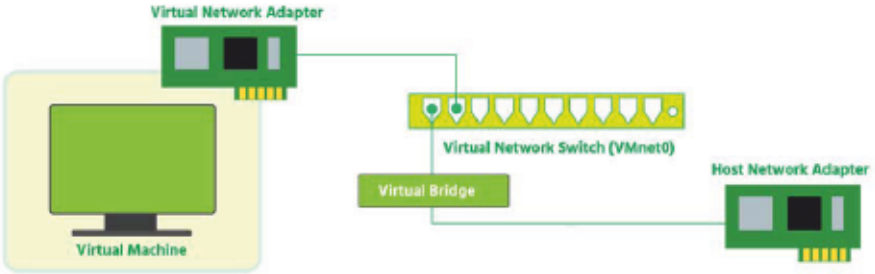

Figure 11. Bridged connection between host machine and virtual machine [2]

There are a few reasons why we opted for this virtual machine configuration. The main reason is that this way the whole system can be run on a single device instead of two. ROS can be run either on a native Linux operating system or a virtual machine, whichever way is preferred. The reason there even is a need for two operating system configuration comes down to the technology choices that have been made previously: Oculus and ROS. Oculus can not be reliably used with Linux since it is not fully supported, and the same thing goes for ROS with Windows. This essentially meant that we were forced to use two separate operating systems to join the session with a robot.

[2]Picture: Sarthak Varshney, 2020, NVC_4.1-1, PNG, accessed 19.2.2022, URL: https://www.c-sharpcorner.com/article/what-is-virtual-networking2/

5. EVALUATION

Since our project did not include designing and conducting an actual HRI study, the evaluation of this project will focus on software testing, the tool and its features, and its development process. In this section we will explain the evaluation plan along with the methods and results of the tests.

5.1. Evaluation Plan

This project does not carry out any data gathering or research itself, but creates a tool and means for it to be done in the future in other research projects. The evaluation for this implementation will be mainly done through the means of software quality and testing. Iterative peer testing has been done since the start of the project for each implemented feature. At this stage, the plan is to carry out thorough testing to test multiple functionalities simultaneously to ensure the wanted functionalities of the project work as indented together on a larger scale.

Testing will include two black-box testing scenarios and one latency measurement. Black-box testing means evaluating a system based on inputs and outputs, rather than focusing on the internal structures. In the rst test, the goal is to con rm the functionality of the movement system and the data gathering system. The test involves one robot avatar and one human avatar. The robot starts from position 1 and the human from position 2. Goal is to switch places between participants by moving the character. Data is collected from this operation and the data points of start and end positions along with the route taken are checked so that the path is logical and the start/endpoints match each other respectively. The second scenario aims to further ensure the data validity with a planned path test. A participant is shown a picture of a path through the Tellus. Their task is to follow this path. Afterward, the path taken by the participant is plotted and compared to the original. If the user makes no mistakes, the plotted path should resemble the planned path. The third test focuses on latency. Latency measurement is done by writing a separate test function to compare timestamp values between local and remote characters.

5.2. Data Validity

The validity of the gathered data was determined with two tests. The rst test was a position exchange test shown in Figure 13. The goal was to create an empirical test to evaluate synchronous movement and the feasibility of the data from the interaction manually. In the test, robot and human avatars started by facing each other in front of the poles seen in Figure 13a. The task was to change the positions to match the situation in Figure 13b. This action shall be interpretable from the dataset. In the gathered data in Figure 13c, the test subjects are highlighted in blue at starting positions and in orange at nal positions. The human avatar highlighted in white is an observer who is not part of the test. As seen from the data, the human start position and the robot end position are nearly the same as in the picture, and vice versa. Paths are also plotted with Excel using the gathered data in Figure 12. The human s path is plotted

in blue and the robot s path is in orange. The robot s start position and human end position differ by a small margin due to the fact that the robot s movement is clunkier and therefore it cannot get as close to the pole as a human avatar.

Figure 12. Position swap test. Robot s path plotted in orange, human in blue.

(a) Starting positions.

(b) Ending positions.

Time	ID	posX	posZ	rotY	robotCamY
3629136,039	player_3_human	-8,8	-4,6	51,4	-
3629136,039	player_2_human	-8,2	-5,7	341,1	-
3629136,039	player_1_robot	-8,3	4,7	212,9	-
3629136,541	player_3_human	-8,3	-4,4	49,2	-
3629136,541	player_2_human	-8,2	-5,7	351,3	-
3629136,541	player_1_robot	-8,3	4,7	212,9	-
3629137,043	player_3_human	-7,9	-4,2	50,1	-
3629137,043	player_2_human	-8,2	-5,7	354,9	-
3629137,043	player_1_robot	-8,3	4,7	212,9	-
Exchange positions...					
3629347,037	player_3_human	-16,4	-1,6	87,3	-
3629347,037	player_2_human	-8,5	5,2	202,3	-
3629347,037	player_1_robot	-8,1	-5,4	312,9	-
3629347,539	player_3_human	-16,4	-1,6	335,5	-
3629347,539	player_2_human	-8,5	5,2	202,3	-
3629347,539	player_1_robot	-8,1	-5,4	311,9	-
3629348,054	player_3_human	-16,4	-1,6	335,5	-
3629348,054	player_2_human	-8,5	5,2	202,6	-
3629348,054	player_1_robot	-8,1	-5,4	311,9	-

(c) Data snippets from start and end of the test.

Figure 13. Simple position swap -test scheme for testing the movement and data collection system.

In the second test, the planned path test, a participant was presented with a map containing a path. The participant s task was then to walk along this path in the virtual reality environment. The path taken by the participant was then plotted in Excel using the gathered data to compare the path to the plan. The plotted path should resemble the original if there are no user errors. All this is demonstrated in the Figure 14. As seen from the picture, the plotted path resembles the original greatly, which further implicates that the data gathering functionalities are working as intended. The differences between the plotted and the planned paths are minimal and the outcome of natural user errors. This is a reasonable indicator that the data collection functionalities are valid in terms of tracking the participants movements in the virtual environment.

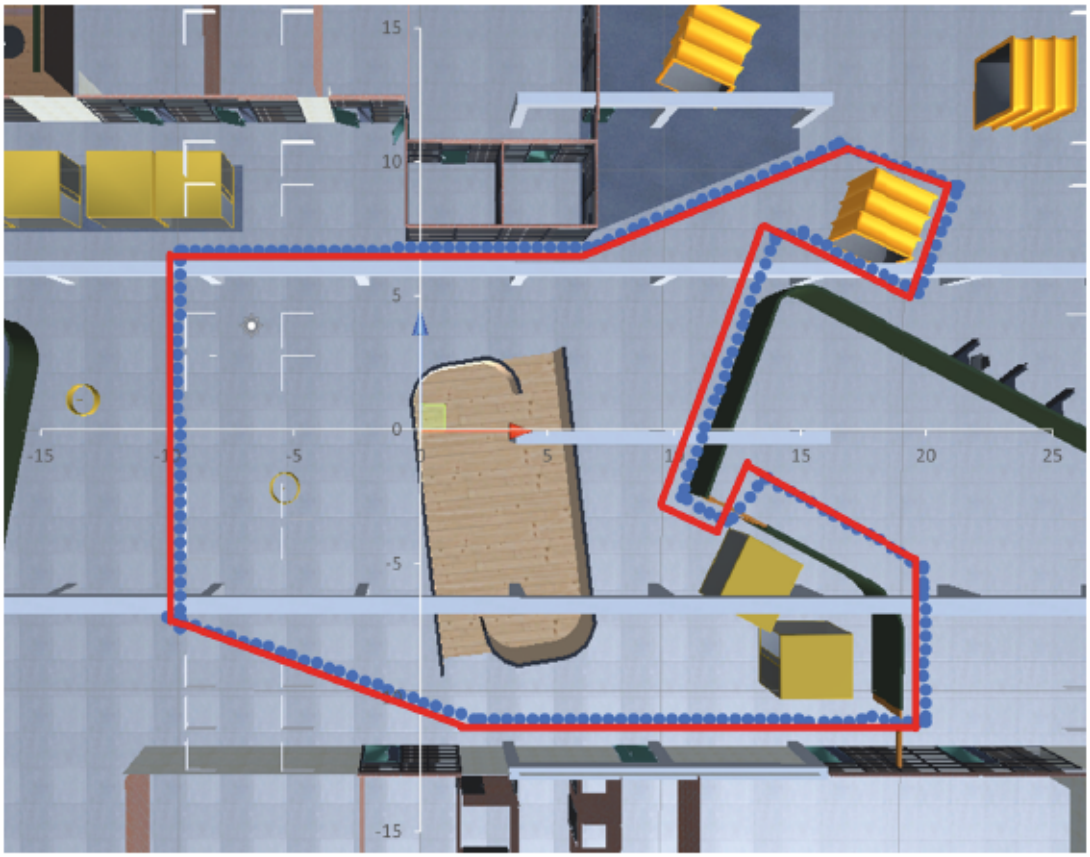

Figure 14. Planned test route drawn in red. Actual travelled route visualized with blue dots. Path plotted from the collected data using Excel.

5.3. Latency

The Round-Trip-Time (RTT) outputted by the Photon PUNs European servers outside the game in Unity Editor is between 34ms and 54ms on development machines. As there is no direct communication between the clients due to the client-to-server architecture, the actual latencies in-game are affected by processing delays between sender, server, and receivers. The in-game latencies were determined by adding a short supporting test functionality to the TrajectoryCollector.cs -script. The test condition was set to be triggered when the threshold coordinate value is exceeded by a player.

When this happens, the test writes the exact timestamp of this action from each client s point of view to their respective local destination le. By comparing these timestamp values, the latencies between local and remote clients can be determined in Table 1.

The test was implemented using three participants and three iterations, each participant being in one role once. The tester-column refers to the character triggering the test condition. This is a local action for the tester and therefore the timestamp, referred to as ts in the chart, is the lowest in the tester column. The timestamp value is further explained in Chapter 4.3 Data Collection. The viewer and reference columns represent remote clients within the same session. Both columns contain the same data: the time the information arrived at the receiving end. From this, the latency between local and remote actions can be calculated. The latency is calculated in the Δtester-viewer column. The difference between the two remote clients is given in the Δviewer-reference -column.

Table 1. Latencies

	tester ts(s)	viewer ts(s)	reference ts(s)	Δviewer-tester(ms)	Δviewer-reference(ms)
1	1403742,965	1403743,163	1403743,166	198,0	-3,0
2	22690,769	22690,945	22690,937	176,0	8,0
3	3563012,022	3563012,199	3563012,195	177,0	4,0

According to the results in Table 1, the in-game latencies settle between 175ms and 200ms. This is nearly one-third of the default sample time of 500ms. In a study completed by Baker et al. [27], a delay of one second as a variable was used to account for communication delays. They chose one second as the delay because delays longer than that can make the control performance signi cantly degraded [28], and it is also in alignment with the delay duration that was used in a study by Luck et al. [29]. From the perspective of positional tracking, the latency of this system is suf cient. The distance in-game a character can travel in 200ms is so short that it does not have a noticeable effect. 200ms is also signi cantly less than the xed delay of 1000ms used by Baker et al. [27]. Orientational tracking is more time-sensitive. A person can rotate their head and hands at a greater pace than they can move around. Virtual reality gear makes the movement and rotations slower and more inconvenient. For this tool, the latency between clients is up to date with the required refresh rates and the latency is not noticeable for the average participant. The Δviewer-reference -column gives insight into how much variation there is between the remote client timestamps. They should be about the same, the data travels through the same components to reach each one. Differences may be caused by the network environment and the performance of the machine itself. The absolute values vary between 3ms and 8ms, values are small and within normal limits.

6. DISCUSSION

Multiple different kinds of tools have been created for HRI research in the past. Many of those tools are speci c to a certain task such as cooking or nding the optimal distance between human and robot interaction. The data gathering system created in this work focuses on the movements and interactions of humans and robots.

Usage of VR offers nearly unlimited possibilities in terms of research environments. It allows the creation of situations and environments that are inaccessible in the real world. It also provides an easy and low-cost solution for data gathering. On the other hand, virtual reality applications can never be as immersive and authentic as real-life experiences. In VR the experience is often lower quality, distorted, and suppressed which can result in a lower feeling of presence and complicate human perception. The characters are also responsible for a great portion of the reality of the environment. The human characters have a cartoon-like appearance to allow realistic movements without heavy animations. The robot is an accurate replica of the real-life version, excluding the stand. One drawback related to the robot is that the rotational Y data of the camera is only visible locally. If the rotation of the camera needs to be collected, the data collection must be enabled on all robot clients to collect the data separately from each user and then combined afterward. These factors are to be considered when analysing the proxemics and trajectories gathered with the system.

The multiplayer functionalities implemented in this system allow a lot of variety and different con gurations with humans and robots. The system is so-called many-to-many, which means that the number of robots and humans is not restricted except for the maximum participant count of the free version of Photon PUN which is 20. This means that the number of robots in an experiment for example can range from 0 to 20 and the same goes for humans. The uncertainty of whether the robot is autonomous or teleoperated allows researchers to create multiple different types of research situations and research different aspects of human behaviour.

The usage of third-party software and hardware has both advantages and disadvantages. If the support for a third-party software is terminated, it can be drastic for the usability of our system and it might require major rebuilding. Then again, the third-party ownership provides outsourced maintenance for it. Oculus compatibility was a requirement for this project and to keep the system as lightweight as possible we only build support for Oculus headsets. At the time of writing this report, Oculus Link requires a wired USB-C connection for a certi ed connection. This essentially means that the wired connection restricts the movement quite a lot, which is why movement with a joystick was also enabled for human avatars. This weakens the feeling of authentic movement but is necessary to allow freedom of movement. Oculus Air Link seems promising to solve the issues involved with a wired connection, but it is still in development.

Oculus is not fully supported with Linux and ROS 2 requires a Linux-based operating system which means that when using Oculus the system requires two operating systems. To avoid using two computers, a virtual machine con guration was used to run the ROS from a native Windows 10. This way we were able to minimize the need for hardware. Regardless of all the challenges, the system can be considered lightweight compared to some other tools used in HRI research.

6.1. Reflection

In relation to the initial target setting, the system meets all the requirements and no compromises had to be made during the design or the implementation. The system supports virtual reality, multiplayer, ROS, and data gathering. It is lightweight and modular to a certain extent. Latencies and data accuracy are at a level at which actual HRI research can be conducted. One of the major decisions made in the design was the introduction of Photon PUN as the service provider for the multiplayer functionalities. Other important decisions included the gathering of the data to a local le and the utilization of a virtual machine to run the ROS on. Any of these decisions could have been a deal-breaker in terms of the usability or functionality of the system, but even after the evaluation and testing, they can still all be justi ed given the requirements.

In relation to the state-of-the-art projects, our system has a lot in common with SIGVERSE, which is a state-of-the-art platform closely related to ours, as stated in 2.2. They both share many of the main technologies such as Unity and ROS, only the versions may differ. The main difference between the two platforms is that SIGVERSE has a lot more features and more diverse data, whereas our system is focused on the trajectories and orientations of humans and robots, which makes it more lightweight as well. This essentially means that the systems have quite different use cases in terms of the studies that can or are reasonable to be conducted. The studies that are conducted on SIGVERSE are more complex and diverse, compared to the ones conducted on our system.

6.2. Future Work

The next step for this tool is to start doing HRI experiments. Studies can be done to research the interactions of humans and robots in multiplayer situations. As changing the experiment environment or deploying multiple autonomous robots does not require a great workload, there are multiple possibilities when it comes to different experiment con gurations. For example, one interesting study could be where multiple robots and humans perform some tasks that require moving across a room and the subject was to make their way from one side of the room to another whilst the robots and humans are moving as well. In this situation, some of the robots could be autonomous and some teleoperated and that information can be used to analyse if the subject s behaviour changes depending on who is controlling the robots. Another possible study that could be organized could be as described: An egg hunt with robots and humans, the rst one to nd all X, wins. In this situation, both teleoperated and autonomous robots could be used to nd differences in human behaviour when competing against humans and teleoperated and autonomous robots.

Some improvements for the system that could be considered are a remote database to collect the data, the addition of spatial sound, sending the orientational information about the teleoperated robot s VR headset over the network, and the addition of more realistic human avatars. The addition of the database would make the data to be more uniform as all the data would be from remote clients and the latencies would all be similar. The spatial sound, in terms of objects making sounds when they are dropped, for example, would make the environment more immersive but it can be a laborious

task to implement. The addition of the talking functionalities would also allow a directional sound to each participant which could enhance the experience in group discussions, but it does not solve all practical issues regarding communications. If the participants are joining the experiment remotely, a third-party communication software is still required to give the participant instructions, etc. And if all the participants are in the same room, then there is no need for communication functionalities in the first place. Making the avatars to be more realistic, meaning that they would have legs and arms and be generally more life-like, has its drawbacks as well. Making the more realistic avatars look natural while moving requires a lot of animating and those animations can reduce the performance of the system. Different kinds of robots could also be added to the system if needed. In terms of multiplayer scalability, another service provider or a paid version of Photon PUN is needed if experiments of over 20 participants on the server simultaneously are to be organized.

From a technical and third-party software point of view, the future of the tool looks promising. Oculus Air Link and ROS 2 for Windows are both long in development and both could bring great quality-of-life as well as actual performance improvements. Wireless high-speed connection to a PC from the VR headset is especially something that has an effect on what studies can be organized. ROS 2 for Windows just means that there is no need for two operating systems anymore which improves performance by making the tool even more lightweight.

7. CONCLUSION

In this work, a human-robot interaction research data-gathering tool was presented. The platform is implemented as a virtual reality multiplayer game. The platform includes human and robot characters. Human avatars are controlled using Oculus virtual reality gear. The robot avatar is a digital version of a real-life robot and is controlled similarly to its counterpart using ROS-middleware. The goal is to create a tool for studying the differences in the participants behaviour when exposed to scenarios where various amounts of humans, human-controlled robots, and autonomously controlled robots navigate around each other. The behavioural analysis focuses on the trajectories chosen by the participants and how the statuses, human or robot and teleoperated or autonomous, affect the behaviour. The main characteristic that sets this tool apart from the other platforms is its lightweightness in addition to the possibility of having multiple robots within the same session. There are no limitations for the combinations of the statuses of the participants as long as the total amount is under the maximum amount supported by the free plan of Photon PUN, 20.

7.1. Contributions

As Juho had a bit more experience with working with Unity than the rest of the group members he was responsible for the initial steps of the Unity project including setting up the environment and working with VR equipment. During the design and implementation stages, Jeremias was main responsible for the multiplayer functionalities and Kalle was main responsible for ROS functionalities. All the group members had equal responsibilities in writing the thesis. Total hours and percentages can be seen in Figure 15 below.

STUDENT SUMMARY				
First name	Last name	Hours	% of project total	% of nominal total (8 cp a' 27 hours ~ 216 hours per student)
Jeremias	Körkkö	256	34.04%	118.52%
Kalle	Veljalainen	250	33.24%	115.74%
Juho	Kurula	246	32.71%	113.89%

Figure 15. Total hours.

8. REFERENCES

[1] Bartneck C., Belpaeme T., Eyssel F., Kanda T., Keijsers M. & Sabanovic S. (2020) Human-Robot Interaction – An Introduction. Cambridge: Cambridge University Press., 6,7 p.

[2] Sheridan T. (2016) Human-Robot Interaction: Status and Challenges. Human Factors 58(4), pp. 525–532. DOI: DOI:https://doi.org/10.1177/0018720816644364.

[3] Kanda T., Sato R., Saiwaki N. & Ishiguro H. (2016) A Two-Month Field Trial in an Elementary School for Long-Term Human-Robot Interaction. IEEE TRANSACTIONS ON ROBOTICS 23(5), pp. 962–971. DOI: DOI:https://doi.org/10.1109/TRO.2007.904904.

[4] Inamura T. & Mizuchi Y. (2021) SIGVerse: A Cloud-Based VR Platform for Research on Multimodal HumanRobot Interaction. Frontiers in Robotics and AI 8(549360), pp. 962–971. DOI: DOI:https://doi.org/10.3389/frobt.2021.549360.

[5] LaValle S.M. (2016) Virtual Reality. Cambridge Univ. Press.

[6] Li R., van Almkerk M., van Waveren S., Carter E. & Leite I. (2019) Comparing human-robot proxemics between virtual reality and the real world. In: 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 431–439. Accessed 19.2.2022.

[7] Yang F., Yin W., Inamura T., Björkman M. & Peters C. (2020) Group Behavior Recognition Using Attention- and Graph-Based Neural Networks. IOS Press, 1626-1633 p.

[8] Koller M., Patten T. & Vincze M. (2022) A new vr kitchen environment for recording well annotated object interaction tasks. In: Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction, HRI 22, IEEE Press, p. 629–633.

[9] Mavrogiannis C., Hutchinson A.M., Macdonald J. Alves-Oliveira P. & Knepper R. (2019) Effects of distinct robot navigation strategies on human behavior in a crowded environment. In: 2019 14th ACM/IEEE international Conference on Human-Robot Interaction(HRI), pp. 421–430. Accessed 1.2.2022.

[10] Gooch A.A. & Willemsen P. (2002) Evaluating space perception in npr immersive environments. In: Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering, NPAR 02, Association for Computing Machinery, New York, NY, USA, p. 105–110. URL: https://doi.org/10.1145/508530.508549.

[11] Messing R. & Durgin F.H. (2005) Distance perception and the visual horizon in head-mounted displays. ACM Trans. Appl. Percept. 2, p. 234–250. URL: https://doi.org/10.1145/1077399.1077403.

[12] Slater M., Usoh M. & Steed A. (1995) Taking steps: The in uence of a walking technique on presence in virtual reality. ACM Trans. Comput.-Hum. Interact. 2, p. 201–219. URL: https://doi.org/10.1145/210079.210084.

[13] Carpinella C.M., Wyman A.B., Perez M.A. & Stroessner S.J. (2017) The robotic social attributes scale (rosas): Development and validation. In: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, HRI 17, Association for Computing Machinery, New York, NY, USA, p. 254–262. URL: https://doi.org/10.1145/2909824.3020208.

[14] Johnson M., Hofmann K., Hutton T. & Bignell D. (2016) The malmo platform for arti cial intelligence experimentation. In: Proceedings of the Twenty-Fifth International Joint Conference on Arti cial Intelligence, IJCAI 16, AAAI Press, p. 4246–4247.

[15] Ros 2 documentation. URL: https://docs.ros.org/en/galactic/index.html. Accessed 2.2.2022.

[16] Konrad A. (2019), Simulation of mobile robots with unity and ros- a case-study and a comparison with gazebo. URL: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1334348&dswid=-8046.

[17] Mathworks documentation. URL: https://se.mathworks.com/help/ros/ug/connect-to-a-ros-2-network.html. Accessed 2.2.2022.

[18] The proven data connectivity standard for the industrial internet of things. URL: https://www.dds-foundation.org/what-is-dds-3/. Accessed 2.2.2022.

[19] Juliani A., Vincent-Pierre B., Teng E., Cohen A., Harper J., Elion C., Goy C., Gao Y., Henry H., Mattar M. & Lange D. (2020), Unity: A general platform for intelligent agents. URL: https://arxiv.org/abs/1809.02627. Accessed 8.2.2022.

[20] Mattingly W.A., Chang D.j., Paris R., Smith N., Blevins J. & Ouyang M. (2012) Robot design using unity for computer games and robotic simulations. In: 2012 17th International Conference on Computer Games (CGAMES), pp. 56–59. Accessed 16.2.2022.

[21] Unity user manual 2020.2. URL: https://docs.unity3d.com/2020.2/Documentation/Manual/index.html. Accessed 8.2.2022.

[22] Facebook to acquire oculus. URL: https://about.fb.com/news/2014/03/facebook-to-acquire-oculus/. Accessed 10.2.2022.

[23] Oculus documentation. URL: https://developer.oculus.com/resources/oculus-device-specs/. Accessed 10.2.2022.

[24] Avramelos V., Praeter J., Wallendael G. & Lambert P. (2020) Random access prediction structures for light eld video coding with mv-hevc. Multimedia Tools and Applications 79.

[25] Photon pun documentation. URL: https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro. Accessed 8.2.2022.

[26] Microsoft documentation what is dll. URL: https://docs.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library. Accessed 8.2.2022.

[27] Baker G., Bridgwater T., Bremner P. & Giuliani M. (2020), Towards an immersive user interface for waypoint navigation of a mobile robot.

[28] Chen J.Y.C., Haas E.C. & Barnes M.J. (2007) Human performance issues and user interface design for teleoperated robots. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37, pp. 1231–1245.

[29] Luck J.P., McDermott P.L., Allender L. & Russell D.C. (2006), An investigation of real world control of robotic assets under communication latency. URL: https://doi.org/10.1145/1121241.1121277.