Design and Deployment of Convergent XR Experiences

by

Anton Franzluebbers

(Under the Direction of Kyle Johnsen)

Abstract

This dissertation explores the development of a convergent extended reality (XR) space through several projects spanning multiple domains, including engineering education, scientific visualization, social virtual reality (VR), health, and productivity. The work represents a shift from developing single-use research projects to creating a broader framework of applications with real-world value beyond basic scientific research goals. Each user study detailed in this dissertation builds upon previous work, contributing to the core technologies of a single monolithic social VR space known as *conVRged*.

The first demonstration of this framework involved developing a novel immersive collaborative land surveying experience. A user study with 85 participants evaluated the VR experience's effectiveness in training the use of surveying equipment, which is otherwise expensive and subject to weather conditions. The study found promising results for using this system in engineering curricula, with high levels of student engagement and enthusiasm. This

project also developed foundational technologies for the social VR space used in subsequent projects.

Next, a scientific visualization tool was developed to allow users to visualize and interact with large-scale point clouds in VR. The evaluation of this tool compared novel selection techniques in both VR and desktop environments, demonstrating relative advantages for each. The advanced rendering techniques developed for this project were integrated into *conVRged*, strengthening the productivity use case for the space.

A third project analyzed a core component of each previous project, as well as many larger VR experiences — locomotion. A user study quantified the advantages of techniques used in our previous work and compared them to novel techniques built for a hand-tracked system. This study also demonstrated the effectiveness of conducting a study in a multi-purpose VR world rather than a single-use system.

Finally, a discussion of the individual tools and development practices highlights the lessons learned and design decisions that led to the creation of a cohesive and valuable collection of VR technologies. This work underscores the convergence of a wide array of technologies that have enabled and continue to enable the use of *conVRged* for diverse research goals in the XR space.

INDEX WORDS: [Virtual reality, Mixed reality, Extended reality, Software framework, Research tools, Social virtual reality, Multi-user experiences]

Design and Deployment of Convergent XR Experiences

by

Anton Franzluebbers

B.S., University of Georgia, 2019

B.S.C.S.E., University of Georgia, 2019

A Dissertation Submitted to the Graduate Faculty of the

University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Doctor of Philosophy

Athens, Georgia

2024

Design and Deployment of Convergent XR Experiences

by

Anton Franzluebbers

Major Professor: Kyle Johnsen

Committee: Fred R. Beyette Jr.

Ramviyas Parasuraman

Changying "Charlie" Li

Electronic Version Approved:

Ron Walcott

Dean of the Graduate School

The University of Georgia

August 2024

Dedication

To my family, who has helped and supported me always.

Acknowledgments

I would like to thank everyone involved in the software development and design, participant recruitment, study design, data collection, and testing, without whom this endeavor would not have been possible. Thank you to AJ Tuttle, who was instrumental in the early development of the project. Thank you to Brook Bowers, who always had advice and suggestions throughout the project. Thank you to the rest of the Virtual Experiences Laboratory for their help with testing and support throughout the years. Thank you to Gulfstream, who supported my research. I would also like to thank all of my committee members, especially my major professor and mentor, Kyle Johnsen, who has been a constant source of guidance and support throughout this process, giving so much of his time and energy to help me succeed. I would also like to thank my family, who has supported me at every step. Finally, I would like to thank Another Axiom for encouraging me to finish this journey.

Contents

Chapter 1

Introduction

Collaborative virtual environments (CVEs) have been a topic of interest in the field of virtual reality (VR) for decades. Research into CVEs started as early as 1990[7, 15], and as the hardware to support these systems has become more accessible, social environments have become one of the most popular consumer applications of VR. Both popular social games Rec Room and Gorilla Tag have claimed to achieve in excess of three million monthly active users[125, 39], which is a large part of the six million monthly active users on the entire Meta Quest platform, the largest consumer VR ecosystem[77]. The need for research into both the development of these social extended reality (XR) systems as well as the evaluation of topics that necessitate the use of CVEs as a platform highlights the need for an increase in development efficiency when creating CVEs. This dissertation aims to contribute to the field by developing a set of tools that can be used across a variety of research domains, with a focus on the development of a single, cohesive social VR space known as *conVRged*.

There are few existing bodies of work that present parallel developments and investigations into the social XR framework space. ARENA is a platform for building collaborative XR applications specifically within the context of WebXR in browsers[92]. ARENA, or "AR Edge Networking Architecture," contains mechanisms for high-bandwidth content distribution with token-based access control. Specific emphasis is placed on the AR use case of shared object placement in the real world, with a variety of plugins such as Bluetooth low energy (BLE) beacons, GPS coordinates, QR codes, and AprilTags available to attach virtual objects to real-world locations. Networking in ARENA is achieved by leveraging the MQTT Mosquitto broker, with a system to bridge clusters of brokers to enable horizontal scaling[80]. A persistent data store is added to allow objects to be retrieved from the pub-sub system across sessions, and an access control system is used between the pub-sub system and the clients to verify permissions. Applications built with ARENA include a VR video conferencing system that leverages Jitsi[33], an AR authoring tool to publish shared world-space 3D primitives, a distributed facial tracking system for avatars, and an XR telepresence system with mixed VR and AR users. Performance evaluations found the pub-sub server was able to handle 160 clients before becoming CPU-bound with a single broker, but can be expanded by bridging multiple brokers. An application of ARENA is discussed in the work by Dasari et al., which leverages ARENA and Jitsi to selectively prioritize certain client pairs in a large VR environment containing traditional video screens[25].

One important collection of work in this space is the Ubiq project[35]. Developed by a group of researchers at University College London, Ubiq is a social VR (SVR) toolkit that has prebuilt features such as avatars, voice, and network connection management. In

contrast to many similar networking solutions on the market, Ubiq allows flexible networking architectures, with either a peer-to-peer mesh or a central relay server. Messages are passed between components in the corresponding network scenes, with each component capable of choosing to exchange messages asymmetrically, implementing a client-server model. However, in order to communicate over the public internet, a public relay server must be used, which passes messages from each of the connected clients to the other peers in the network room. The Ubiq system also provides higher level abstraction layers for XR input by making use of the built-in Unity XR Toolkit[134], animated avatars for a three-point tracking rig for a system that tracks the head and two hands, voice chat through a separate WebRTC communication channel, structured event logging that can be collected on one of the peers, and a rooms system to manage the connection of sets of peers to each other with three-digit codes. Performance evaluations of the system found that the server could handle 50 users before significant increases in latency occurred, and a desktop client could handle 30 peers before the frame rate dropped below 60 frames per second. The later work Ubiq-exp demonstrates a set of tools that facilitate the running of remote experiments, with three pilot experiments serving as demonstrations of the systems capabilities[109]. Ubiq has also been used in teaching scenarios, as detailed in Friston et al.[34]. This work discusses the tradeoffs of using an in-house networking to teach a VR course, and concludes with promising results for the advantages of Ubiq's networking architecture and open-source design. Another work by Numan et al. leverages Ubiq to integrate external AI frameworks into an SVR environment by offloading the work to a client with sufficient computing power to run image generation models or a large language model (LLM)[85].

Beyond research applications, social VR games are one of the most popular real-world uses of XR hardware. Rec Room is a social space that focuses on high quality first-party experiences such as virtual sports or quests, with the ability for users to create and share worlds, customize their avatars, and participate in scheduled events[98]. User-generated content in Rec Room can be achieved using a custom visual scripting system. This system handles the synchronization of shared objects through the custom networking system, and highlights areas of the code that are causing slow execution. However, Rec Room does not contain a variety of tools necessary for a research platform, such as the ability to export data through a logging system, deeper customization of avatar visualization, and the ability to configure deeper aspects of the system. VRChat is a similar commercial social VR platform, with a greater focus on user-generated avatars and worlds[126]. VRChat supports asymmetrical platform interaction, with support for both VR and non-VR users in the same environment. This hybrid approach makes concessions as to the affordances of remote users, as non-VR users are not immersed in the same way and do not have the detailed hand, controller, or even body tracking of VR users. On the other end of the spectrum, Gorilla Tag focuses entirely on first-party content, with a unique locomotion system that drives the gameplay[38]. Gorilla Tag uses Photon PUN for networking[94], and drives several non-realtime social features through Microsoft's PlayFab platform[78]. While much can be learned from the success and design of these popular applications, they are generally not suitable for VR research applications due to the lack of control over the environment.

This dissertation starts with a description of a system built for collaborative training of land surveying equipment in VR. This application built the foundation upon which the

subsequent work of the next few years in this dissertation was built. While the application was built from the ground up as a single-use system, specific care was taken to ensure that the application was usable in an unsupervised setting, with features that were not strictly necessary for a guided user study. In fact, the design of the system as an application first and research apparatus second enabled some unique interactions during the study that would not have been possible otherwise. The problem of communicating with a user immersed in a VR headset is well known, with many projects aiming to provide creative solutions to bring real-world context or people into the virtual world of the user, a problem known as bystander awareness [86, 59, 87, 70]. Another solution to this problem is to meet the users where they are by having the experimenter also wear a VR headset. This proved to be useful when guiding the majority of the 85 participants who had no experience with a VR headset or controllers, as the experimenter could point, guide, and manipulate objects in the virtual world during the training phase to introduce the participants to the system. As detailed in the chapter, the experiment was conducted both with dyads and individual participants; however, this advantage was seen in both cases, demonstrating the benefit of a networked environment even in research situations involving a single participant. While the land surveying application was not our first project in this line of work that involved the creation of VR interaction, locomotion, and management systems, it was the first where an attempt was made to develop these systems in a reusable and modular way.

After the engineering education study, I began work on a separate project called "Virtual Family Room" (VFR), which was funded with the goal of creating and studying a social virtual environment for deployed military parents to interact with their children and family

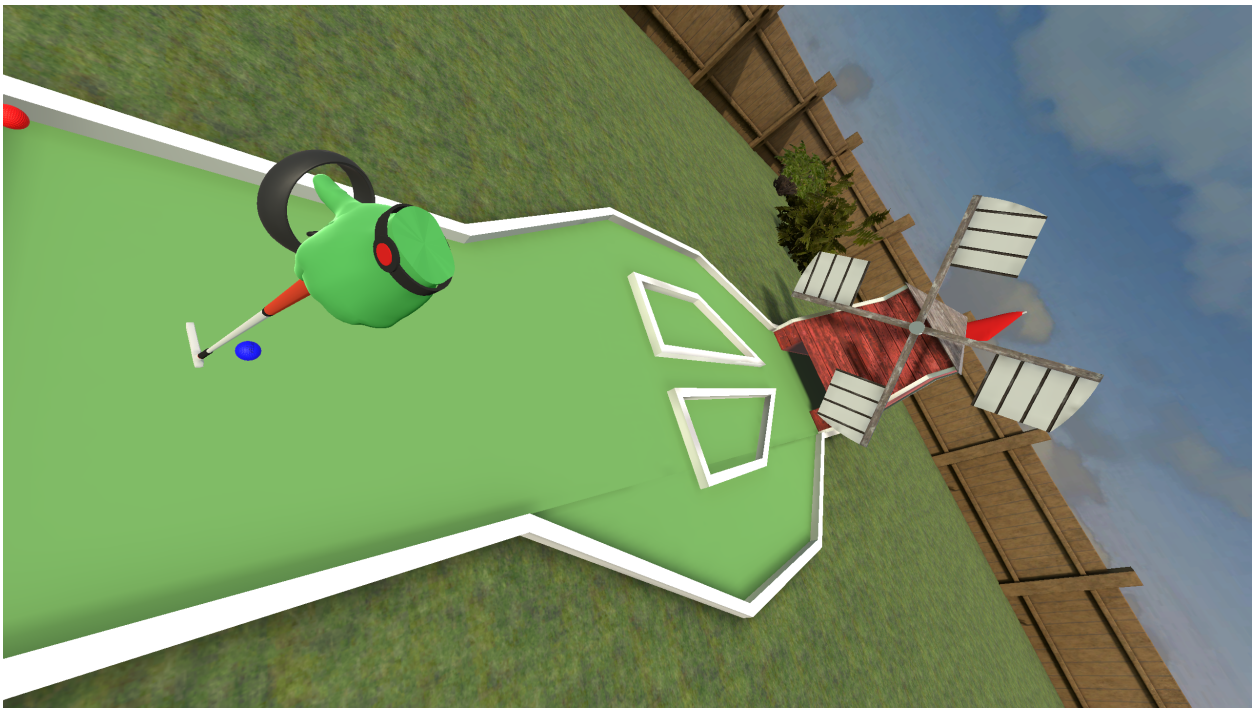

Figure 1.1: A screenshot from within the *VFR* environment demonstrating the virtual mini-golf course.

Figure 1.2: One of the mini-games in the *VFR* environment was an area to collaboratively control miniature cars.

remotely. While the results of this project are not detailed in this dissertation because the primary research goals were conducted by another student, the software development effort during this period was mine and was instrumental in turning the single-use land surveying tool into a multipurpose application. The goals of this new project as an entertaining social space rather than an educational tool led to the development of a multitude of smaller activities, such as remote-controlled cars, basketball, a mini-golf course, and a checkers board, as seen in Figures 1.1 and 1.2. The modularity of the drawing tool developed for the land surveying tool was proven with its reuse in *VFR*, as seen in Figure 1.3.

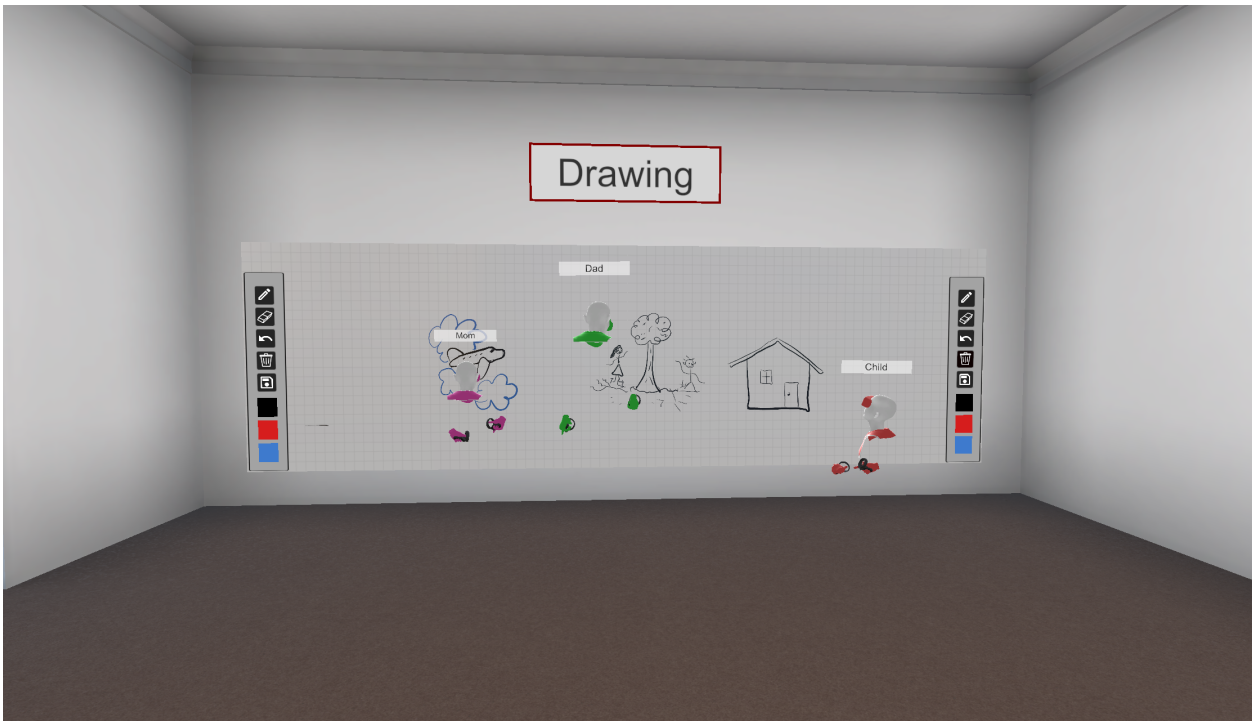

Figure 1.3: The drawing tools first developed in the engineering education application were re-used in the *VFR* environment. What was previously a productive tool for marking down data points became a collaborative creative expression tool.

Another pivotal development in the progress of this project was the submission and approval of *VFR* to Meta's App Lab store. As this application was moving outside the realm of a controlled research environment, distribution of the application and the automatic deployment of updates with fixes and new features were critical. Meta's standalone Quest platform was also essential to the deployment of this application to real users, as it was the only widespread, cost-effective mobile VR headset at the time. Remote participants in our study could install the application through the publicly accessible but unlisted store page shown in Figure 1.4. One lesson learned from the process of app store submission was our initial rejection due to the inclusion due to the perception that it was targeted towards children despite no supporting metadata approving the use of the application by children. This was likely due to the inclusion of the word "family" in the title. We took this opportunity to rebrand the application to *conVRged*, also signifying the multipurpose use of this application as it began to be used for other projects. Meta has since changed their stance on this issue, with support for child accounts [17].

Around this time, work began on a productivity-focused virtual workspace that pulled in documents and files and allowed the user to arrange them spatially. We called this project *DataFoldvr*. While this project as a whole is not described in this dissertation, one particular tool that was part of the larger application was a point cloud annotation tool. The research goals and results are described in Chapter 3. While not a primary research goal, the locomotion technique used in that study was a grab-move method that we felt suited the application very well. As described in the chapter, participants did not have difficulty adjusting to the smooth-motion technique, and this encouraged us to add it to the

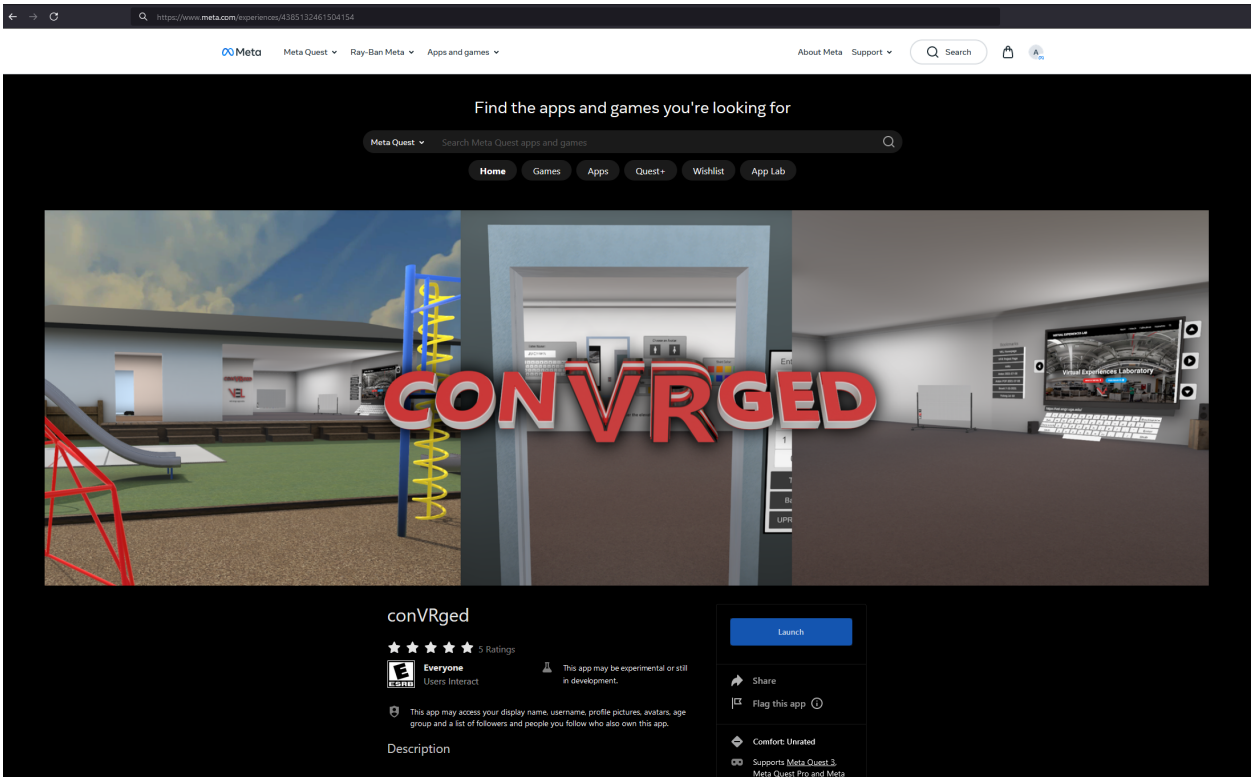

Figure 1.4: The Meta App Lab store page for conVRged. This enabled easy distribution of the application and its updates to internal and external users

set of standard techniques of the version of *VFR/conVRged* at the time. Only later was this technique compared and analyzed with more scientific rigor, as detailed in Chapter 4.

When the global pandemic caused research labs to work from home, our lab was in a unique position for the real-world use of our own software, a practice known as "dogfooding [45]." Three times a week, the members of our lab put on their VR headsets and met in *conVRged* to discuss plans, present papers they had read, and test features of the system. This time was incredibly helpful to the development of the virtual environment and resulted in a large leap forward in the quality and usability of the system. Trivial but sometimes frustrating bugs were fixed much more quickly, as they were affecting my daily life directly rather than perhaps having a negligible impact on a user study in the future. While it is not my hope that the conditions under which this testing happened occur again, we do encourage other labs to engage in the rigorous, long-term testing of their multi-user virtual environments as we did.

The fourth chapter of this dissertation documents the user study evaluating the locomotion systems used in *conVRged* for both a hand-tracked and controller-tracked condition, as well as presenting the most complete version of the social VR space that has been developed across these projects. Locomotion had been a critical part of the application since the beginning, but this work quantified the relative benefits of the traditional teleporting system and the newer grab-move technique first added as a result of the work in Chapter 3. The work in this chapter also introduces and compares a novel set of locomotion techniques for free-hand systems following the same two primary modalities of teleport and grab-move. Again, as this software was not specifically developed for a single study, the systems were developed to

work well in messy real-world situations where the focus of the user is not on the locomotion system but rather a conversation or other manipulation task. The operation of this study also proved the utility of *conVRged* as a study tool. Outside of the addition of the free-hand techniques to the application, very few modifications to the system's code were required to support the running of the user study. The study was conducted much as a user study in the real world would, with an experimenter in VR guiding the participant through each task by means of demonstration or gestures with their avatar. Some logging was added for task-specific objectives, but many of the output measures analyzed in this work were already instrumented in the system.

The fifth chapter of this dissertation discusses the overarching development of conVRged, highlighting the lessons learned, design decisions, and the integration of technologies across projects. This chapter focuses on the value of the individual components, all of which are publicly accessible and can be used in third-party projects. These components represent the longest-lasting software contribution of this work, with many of them already being used in projects unaffiliated with my own work. This chapter also details the challenges faced in some of our projects, such as the lack of availability of an internet connection, and how these challenges were overcome with the self-hosted infrastructure that we built. All of these components are part of the cohesive whole in *conVRged*, a multi-purpose application that is being used by various projects into the future.

1.1 Chapters

1.1.1 Chapter 2

The work in this chapter was published in the proceedings of the 17th International Conference on Remote Engineering and Virtual Instrumentation (REV 2020)[31] and investigated the design and development of a engineering education application designed to teach land surveying in VR.

Personal contributions

I developed a majority of the software architecture and components for the application, including many of the design decisions that accompanied them. I also was the sole developer of the simulated equipment, including the 3D modeling of the objects as well as emulating the behavior of the equipment's software. Along with AJ Tuttle, I designed and conducted the user study and conducted data analysis of the results.

Collaborators

Undergraduate student AJ Tuttle assisted in the software development for many parts of the application as well as in the conduction of the user study. Particularly, the virtual drawing surface was developed primarily by AJ, and the initial versions of the networking system were developed with approximately equal parts between myself and AJ. Robert Baffour assisted in the recruitment of participants of the user study and as reference for the educational

materials. Kyle Johnsen provided guidance in an advisement capacity and assisted with some design decisions and data analysis.

Relevance to dissertation

This work built the foundation of the collaborative multi-user system that is continued in the rest of the dissertation. While they were not in their final form, the interaction, locomotion, and networking systems used in subsequent projects were first explored and developed in this work. This work also highlights an important facet of the collaborative XR environment presented in this dissertation - education.

1.1.2 Chapter 3

The work in this chapter was published in the proceedings of the 2022 ACM Symposium on Spatial User Interaction (SUI 2022)[32] and compared a immersive VR point cloud annotation and rendering system to a 2D-desktop system. This material is based upon work supported by the National Science Foundation under Grant No. 1934481.

Personal contributions

I was the sole developer of the application presented here, including the loading of point cloud data, the development of the rendering techniques described in this work, and the annotation methods. I conducted the user study and performed the data analysis.

Collaborators

Changying Li's lab provided point cloud data and insight for their needs. Kyle Johnsen assisted in an advisement capacity throughout.

Relevance to dissertation

While the software used in this work is not collaborative, the point cloud rendering techniques developed here were added to the *conVRged* application to enable productive discussions in VR regarding the data. The work described here was funded by an NSF grant for convergence research, which ultimately led to the first cross-lab use of *conVRged* during this project.

1.1.3 Chapter 4

The work in this chapter was published in the proceedings of the 29th ACM Symposium on Virtual Reality Software and Technology (VRST 2023)[30] and compared hand-tracked and controller-tracked locomotion systems for a variety of tasks.

Personal contributions

I was the primary developer of the application presented in this work. I designed and conducted the user study and performed the data analysis.

Collaborators

Kyle Johnsen contributed to the development of the networking system used in the application as well as the menu interaction system used for the in-application questionnaires. Kyle Johnsen also assisted in an advisement capacity throughout.

Relevance to dissertation

The application presented here represents the culmination of all of the prior studies and applications. While the user study focuses on locomotion methods, which are a significant component of the social VR world, the work also demonstrates the capabilities of *conVRged* as a multi-purpose framework for a diverse set of research goals.

Chapter 2

Collaborative Virtual Reality Training Experience for Engineering Land Surveying[1]

[1]Franzluebbers, Anton and Tuttle, Alexander James and Johnsen, Kyle and Durham, Stephan and Baffour, Robert. 2021. Cross Reality and Data Science in Engineering: Proceedings of the 17th International Conference on Remote Engineering and Virtual Instrumentation 17. 411-426. Reprinted here with permission of the publisher.

2.1 Abstract

Training students to perform accurate and efficient land surveys through hands-on laboratory experience takes time, space, and expensive equipment. To alleviate demands on these resources, we designed and implemented an immersive virtual reality training simulator that closely replicates both the equipment and social experience of land surveying. Our pilot study of 85 undergraduate engineering students shows high levels of student enthusiasm for the approach, which, by virtue of the recent affordability of virtual reality equipment, is now feasible for a multitude of use cases throughout the engineering curriculum.

2.2 Introduction

Land surveys are integral parts of a multitude of engineering projects, which rely upon accurate topographic measurements for analysis, design, and construction processes. In practice, land surveying involves the use of a suite of sophisticated measurement technologies, such as the "total station", a composite surveying device used to perform many measurement and computation tasks. It also requires teamwork to handle equipment and perform surveys efficiently. As such, hands-on training experiences currently require significant investment in the requisite technology alongside appropriate space and time management to allow for student teams to practice. This may be practical for a dedicated technical training program, but is burdensome for a undergraduate engineering program, where students may only have a single semester course that introduces basic principles, but are later expected to perform

land surveys and work with land surveyors. Under such constraints, students may have few opportunities to gain hands-on practice and feedback, little or no access after the course, and hence difficulties learning, retaining, and using land surveying skills.

Much like other training situations that feature expensive equipment (e.g. pilot training), land surveying simulators have been proposed as a practice tool that greatly alleviates resource constraints [60, 110]. Both realistic environments and surveying equipment functionality may be simulated effectively. However, their validity has not received much attention. In terms of face validity, an issue with extant simulators has been the use of a desktop/laptop based interface. These interfaces do not replicate the perceptual experience of surveying. To view the virtual environment, a joystick or mouse is used, rather than head motion. A monoscopic view of the environment is presented, reducing depth perception. Interaction with the virtual equipment involves mouse clicks and button presses, reducing the likelihood skills would transfer to real equipment. Furthermore, though multi-user experiences are possible, the interface limits natural, nonverbal communication from being employed.

An oft-proposed solution to the above problems involves the use of more "immersive" virtual reality technologies that more closely replicate real-world training. This would be akin to a full-motion flight simulator; however, until very recently (c.a. 2016), such technology was more expensive and difficult to deploy than real surveying equipment, making it entirely impractical to consider. Both of these issues have been largely addressed. Using affordable technology (as of this writing, $400 USD), this work demonstrates that it is now possible to simulate a first-person, immersive experience in a virtual world to a sufficient level of visual, and interaction fidelity. The common technology consists of a head-mounted display and two

controllers, both of which have their position and rotation accurately tracked with respect to the physical environment. Notably, this can now be achieved without any world-mounted hardware, and without exposed wires, or the requirement of a separate computer, making the technology entirely self-contained and portable, overcoming many previous barriers to adoption (See Section 2.4).

This paper explores the readiness of contemporary virtual reality technology for land surveying training through a pilot study. The study, described in detail in section 4, involved 85 students enrolled in an introductory engineering course. Participants learned to operate a total station device to take accurate measurements. Afterwards, they provided feedback. Results strongly support continued integration of virtual reality technology into land surveying training, as well as integration to further courses that may now provide training where it was difficult or impossible to employ before.

2.3 Related work

Kuo et al first demonstrated the feasibility of Using interactive 3D graphics technology to provide a simulated surveying experience [60]. The tool, SimuSurvey, featured an interface with several control panels that included an exocentric 3D view of the environment and equipment, which could be controlled through a set of associated widgets. A preliminary evaluation showcased the efficiency and effectiveness of SimuSurvey in teaching and learning relative to outdoor laboratories, with high levels of student and instructor enthusiasm for the approach. However, they noted that the interface lacked authenticity with respect to the real

world and that use of real instruments was still necessary for skills training. A follow-up study vastly improved upon this interface through human-centered design practices, optimizing the training tool for the presented tasks, yet the lack of realism in the interface was still noted as a major barrier for users [67]. Further work included the simulation of systematic errors in the manufacturing and calibration of the device [108].

Continued work from this research group developed SimuSurveyX, a serious-game built for the Microsoft XBox gaming console (also compatible with Microsoft Windows systems), releasing it for public use. This system was a notable departure from Windows-Icon-Mouse-Pointer (WIMP) interface used in SimuSurvey to first-person, exploration-based simulation using a game controller. Rather than using abstract widgets to control virtual devices and visualize measurements, these functions were mapped directly to devices and users could more directly perform many setup and control functions. Graphical realism was also greatly improved. Though details on how this system has been used subsequently are sparse, at least one research project has used it to improve an online surveying course [37].

The work presented here extends the approach of SimuSurveyX to focus on interaction fidelity, to not only integrate the visual and functional experience of surveying, but to also allow users to control the surveying instruments in naturalistic ways through motion control. Further, the system allows for multiple simultaneous users to both operate equipment together and to communicate with each other through voice and avatars.

2.4 System Description

2.4.1 Design

Rather than build a dedicated surveying simulation application, as has been previously done, this work instead focused on developing a new education and collaboration platform that could be used for surveying. The major difference between these two approaches is that the interface for a dedicated simulator can be tailored to surveying in all aspects, while a platform must consider how an interface can be used for many potential applications. For example, two prior efforts in using immersive virtual reality interfaces for engineering education have focused on engineering statics as the target application [112, 76]. Integrating such applications together with surveying could provide a stronger justification for using virtual reality throughout an engineering curriculum, such that students would not have to relearn the interface for each application and could navigate between applications from the same program.

The key to this approach is the concept of a reality-based interface [48]. In a reality-based interface, the real-world is the basis for design, both in terms of the simulated world and the user interface. Instead of the objective being to make the system easier to use to accomplish tasks, the goal is to require mimic how those tasks would be performed in the real world. This has two primary advantages. The first is that the simulations can be self-contained, insofar as they are in the real world. For example, a virtual total station that directly mimicked a real total station would use the same interface as the real one, such that its functions were

activated by buttons, and it could be adjusted by rotating components and knobs. Similarly, activating those buttons and turning the knobs would be accomplished through simulated physical collisions. This has the logistical advantage of allowing many simulations to co-exist without requiring adaptation of the user interface to support them.

Enabling this approach, physically immersive virtual reality interfaces simulate the perceptual experience of first-person, hands-on interaction, which further support the phenomenon of mental immersion (presence) in a virtual world [24]. Modern systems have made great strides in supporting reality-based interaction at low cost. The system used in the current work, the Oculus Quest (approximately $400 USD), provides a 1440x1600 pixel resolution display per eye at 72 Hz frame rate, inside-looking out display position and rotation tracking (using pure computer vision from 4 integrated cameras, without pre-defined reference points), two hand-held and tracked (from the same headset cameras), and a built in Android-based computing platform for simulation and rendering, without any external wires. Though some compromises must still be made, primarily due to the lack of high-fidelity haptic (touch) feedback for controls and insufficient resolution of the displays for fine-detail, the overall usability was determined to be high enough to support the current applications.

2.4.2 Platform Features

The platform, built using Unity3D, goes by the name ENGREDUVR (pronounced engr-ed-oov-er, short for ENGineeRing EDUcation in Virtual Reality). It is, first and foremost, built for collaborative learning. To this end, it focuses on two key team-communication supports - avatars that can communicate non-verbally through gestures and drawing surfaces that can

be used to illustrate and take notes. It also supports verbal communication over a network, though in collocated situations, this is turned off in favor of speaking to each other directly.

The avatars, though customizable through the Oculus Avatars platform feature, are intentionally set as generic. However, to enable rapid identification of each person at a distance, one of several colors may be chosen by each user, which color the visualized headset and hands, leaving the rest of the body as white. As only the head and hands have tracked references, a full virtual body is not used (common in most virtual reality games, as simulating a realistic virtual body without it being tracked has little value for non-verbal communication). In addition, the user may select a name that is displayed above their avatar's head. This interface is shown in Fig. 2.1.

The movements of avatars and all virtual objects are synchronized across a network, using a networking platform called Photon Unity Network. Photon enables network synchronization across a wide variety of network topologies, and is cross platform. Each device connects to a public Photon server, which then synchronizes messages across devices. For ENGREDUVR, this allowed support for heterogeneous hardware platforms (as was used in the pilot study, described below), and for the use across large distances without significant regard for how a device is connected to the internet (for example, behind network address translation and firewalls). To control for network bandwidth, avatars are synchronized at a slow update rate (10 Hz), with interpolation used to smooth visual movement.

A mechanism was introduced to allow for collaborative note taking and design. This idea has been explored previously for collaborative problem solving [112], though in that case required the user to have an electronic drawing tablet. As this approach required the user to

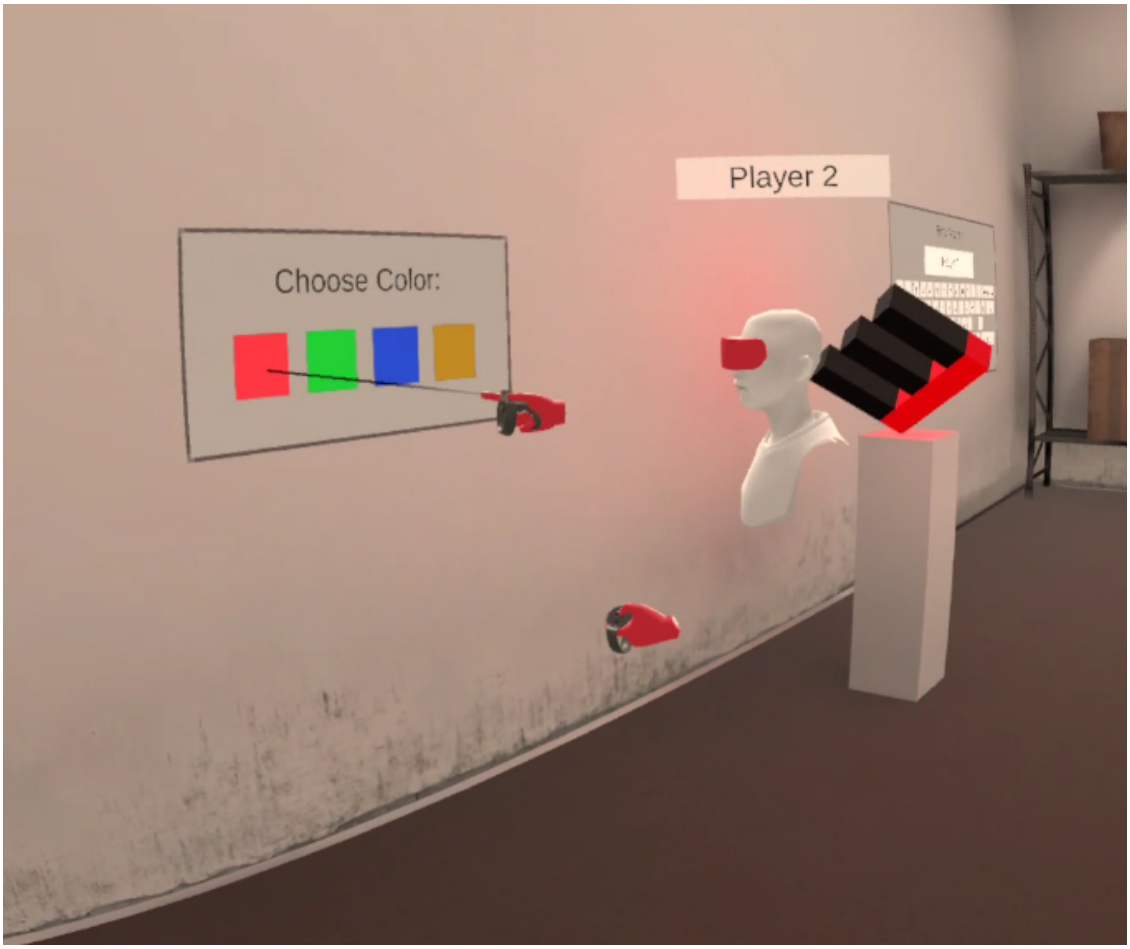

Figure 2.1: A user chooses a coloring for their avatar. Afterwards, they may customize their name using the virtual keyboard in the background.

hold a physical device, inhibiting manipulation of the instruments, we opted to build virtual whiteboards (see Fig. 2.2) that could be drawn to using only the tracking controllers (as is commonly done in virtual reality games). Though lower accuracy, precise diagrams was not strictly necessary for surveying. More writing-centric applications for the ENGREDUVR platform would need to consider the integration of higher accuracy writing instruments, which is possible and complementary to the ability to write and draw using only the controllers. As

with avatars, diagrams are synchronized across the network. However, unlike avatars, these diagrams introduce significant synchronization burden when new users join, as they must be "caught up" to the current state. To minimize this burden, drawings are represented as a sequence of line trajectories (as opposed to bitmaps) that can be efficiently stored and transferred. This approach also has the benefit of allowing for efficient "undo" operations, and enforcing a strict ordering for overlapping lines between users. If two users simultaneously draw lines that overlap each other, after synchronization, they see the same image.

Though most of the interface is reality-based, i.e. it matches that of real-world surveying, including devices and terrain, there are several notable features that must be introduced to users that address the practical limitations of virtual reality.

First, assuming the real world environment in which the virtual reality system is used is significantly smaller than the virtual environment, a travel technique must be introduced. The choice for many extant virtual reality systems is a technique called "raycast teleportation". Using either of the tracked controllers, the user indicates their intent to teleport by pushing up on a thumbstick. This activates a parabolic-shape indicator line that originates from the tracked controller and extends a short distance outward until the line intersects a surface that can be teleported to, as seen in Fig. 2.3. The intersection point is where the user will be teleported to when they release the thumbstick. This enables rapid exploration of the virtual environment without the need to physically walk (though walking is still possible, within the physical space, and does control virtual position). To turn, users may simply turn their physical bodies, or they may tap left or right on the controller thumbstick to "snap-turn"

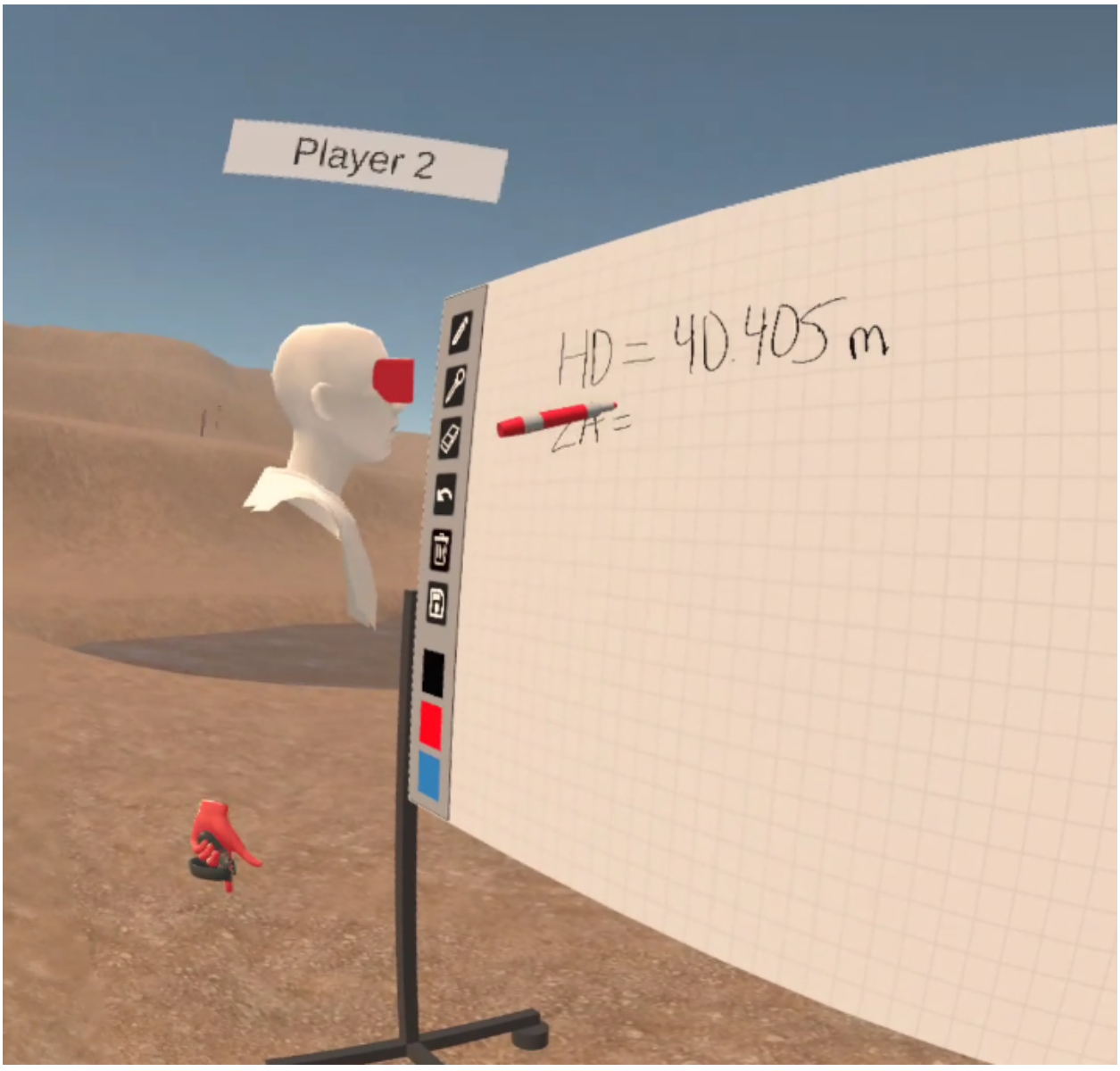

Figure 2.2: A user writes measurements on a virtual whiteboard.

in discrete, 45 degree intervals, with discrete turning preferred over continuous turning to mitigate cybersickness associated with the latter.

Figure 2.3: A user teleports to join their partners. The black line is the arc-raycast technique.

Next, a mechanism was introduced to manipulate the movable components of virtual devices, which lack the tangibility of real devices. When a movable virtual device component is intersected by the user's hand-avatar, it highlights, indicating that it can be manipulated. Upon pulling the index-finger trigger, the movement of the component becomes mapped to the movement of the controller, with constraints set within the component that was grabbed. Reusable code allows the exact specification for different types of components. For example,

small knobs may be rotated by twisting the controller after grabbing. Larger joints, such as the total station body, are turned by moving the controller in an arc. This is only semi-intuitive for new virtual reality users at first, but is quickly learned by practice. Two issues occurred due to hardware limitations. First, the resolution of the display was insufficient for seeing small text (e.g. button labels) at normal distances (text could still be seen by getting very close). To overcome this, when the user brings their hand towards a virtual button, the hand morphs into a small magnifying glass that makes text significantly more visible. Second, the computer vision tracking of the controllers fails when the controllers are very close to the head. This would naturally occur when adjusting knobs on the total station while peering through the optics. As this mechanic was also difficult because of the lack of tactile feedback, we chose to avoid having the user look through the optics, instead creating a virtual viewfinder that allowed the user's head to remain farther away from the total station while manipulating.

Finally, several in-world menus are introduced for simplicity in controlling the less educationally relevant, but still functionally important aspects of the simulation such as avatar choices and navigating between surveying scenes. These were used by pointing a virtual laser line at a menu item and pulling the trigger. The laser line could also be used as a communication device, since it's visibility was shared between users.

2.4.3 Surveying-specific features

Arbitrary virtual landscapes can be surveyed, with some limitations on scene complexity due to the need for a high frame rate that mitigates cyber sickness. As the landscape is

virtual, the exact values of all surface points are known, which allows for rapid specification of assignments with known solutions (e.g. determine the Cartesian position of each landmark). Beyond this, two devices were simulated, a total station and a prism-rod.

The total station was the most complex device to simulate. From a simulation perspective, it encompassed two distinct components, the station itself and a tripod that it could be attached to. Once attached, the station must be manually leveled through a manipulating a sequence of translational and rotational joints. The tripod could be moved to an arbitrary location in the virtual world by grabbing and teleporting. When grabbed, the legs automatically extended such that the feet hit the ground, if possible. Once positioned, the legs could be independently adjusted in length by grabbing and translating. Like a real surveying tripod, a 2D circular bubble level is used to determine the rough orientation of the station platform. Once the tripod is leveled, the station is finely leveled throughout its 360 degree range by adjusting three foot screws in a systematic fashion. Once leveled and positioned, the station can be used to take measurements through course (directly rotating the device and adjusting the pitch) controls, and fine grain controls (rotation knobs). Finally, the device control panel can be used in a nearly identical manner as a real total station to take measurements. The total station and control panel can be seen in Figures 2.4 and 2.5, and the adjustment of the course rotation knobs in Figure 2.6. The prism rod operates similarly (Figure 2.7), with a circular level bubble and automatically adjusting the length of the leg, while turning the prism to face the station.

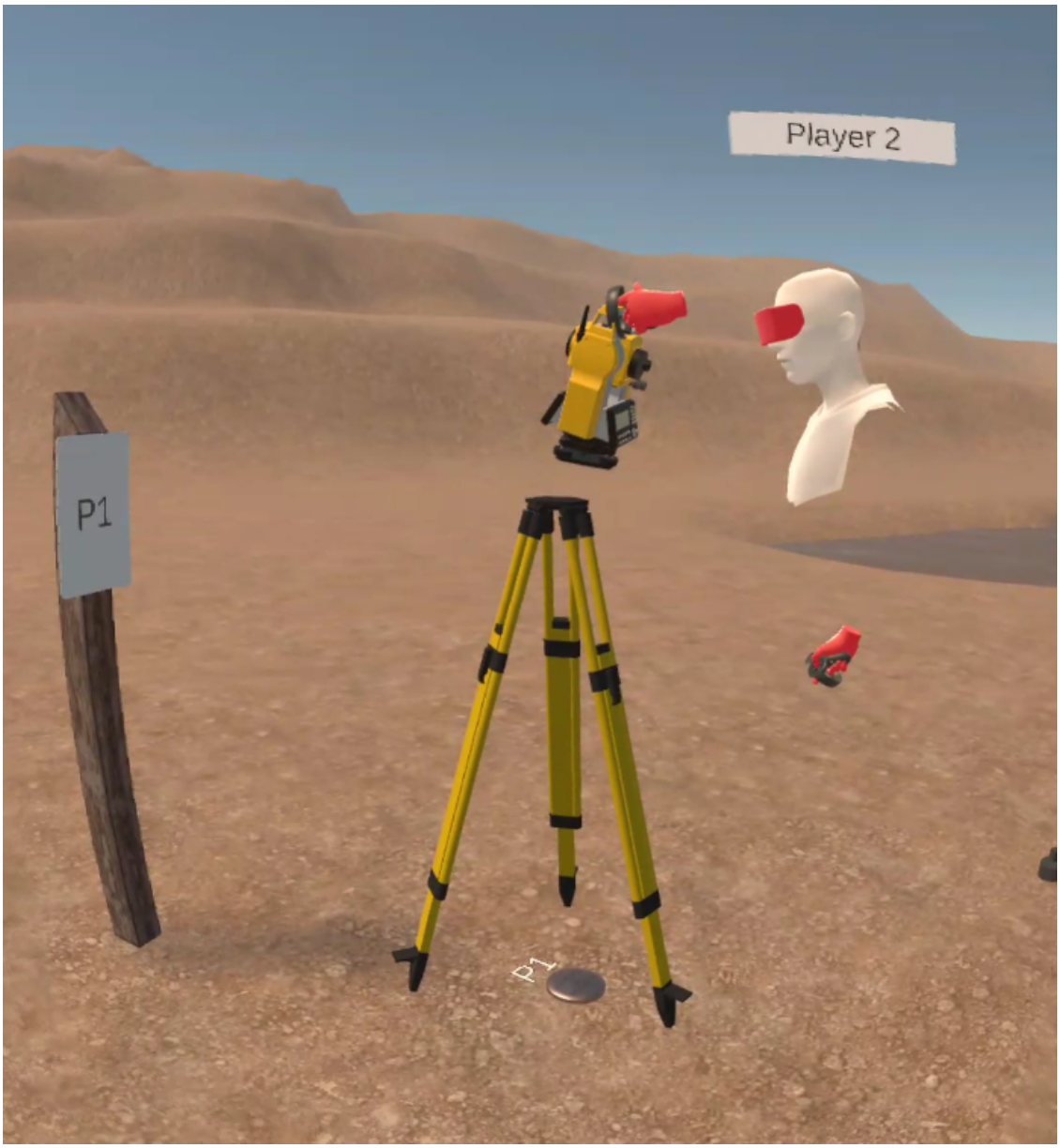

Figure 2.4: Total Station and tripod.

Figure 2.5: Total station control panel.

Figure 2.6: Course rotation of the station.

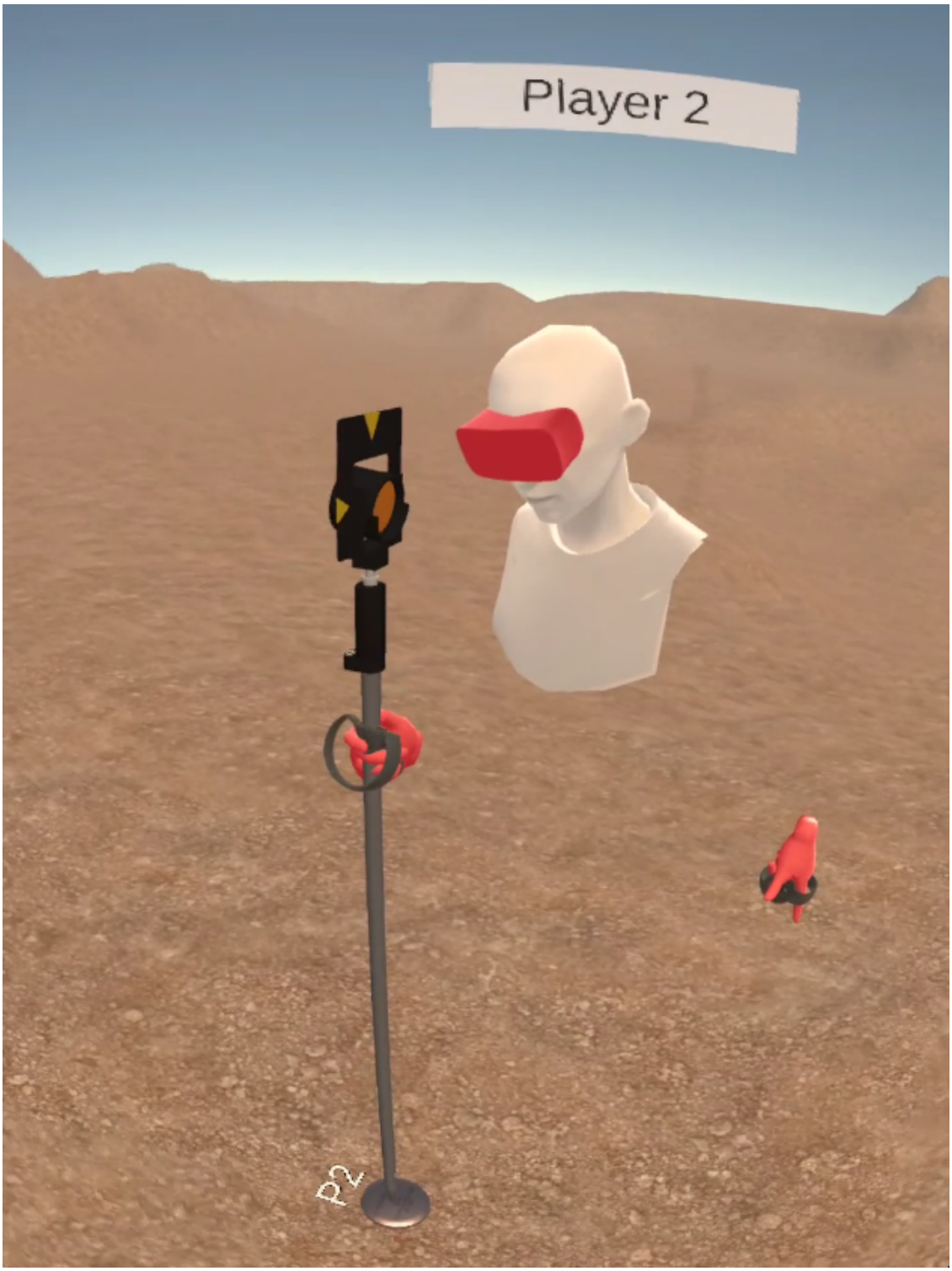

Figure 2.7: Placing the prism rod.

2.5 Study

After Institution Review Board approval, a pilot study was conducted to determine the usability of the immersive virtual reality surveying system from Section 2.4 and to obtain student feedback on the approach, with the intent on using this feedback to prepare for integration into land surveying coursework. In addition, a single independent variable was investigated, whether students performed a test survey in pairs or individually, to mimic how students may use system in practice.

2.5.1 Population and Environment

Given that teaching surveying skills was not the intent of this study, we instead involved an introduction to engineering course to evaluate the technology. The instructor (a coauthor) for this course, who also teaches the land surveying course, created a course assignment that required students to sign up for time slots in pairs of two, where they would visit the laboratory and go through a practice exercise in land surveying. The purpose of the assignment was unrelated to land surveying, but instead was to evaluate the experience of using virtual reality technology in education. This assignment was worth 5% of their course grade, and could be performed without agreeing to participate in the study. To protect students from potential coercion, the instructor was not involved in conducting the experiment and was not made aware of the identity of participants, although all eligible (4 students could not participate due to being under 18 years of age) students chose to participate in the study. In total,

85 eligible participants were recruited from the class of 89 students, with the study being conducted over 3 weeks due to the need to schedule 30 minute sessions in pairs.

Though the technology was capable of working in any indoor space with an internet connection (or without, if there is only one user), to control for external factors, the study was conducted in a dedicated virtual reality laboratory that had ample space for multiple collocated participants to spread out (See Fig. 2.3, lower left). As discussed earlier, by collocating participants, they could speak to each other directly, without using microphones.

The virtual environment used for the study was designed to look like a quarry or building site, and is shown in Fig. 2.8. This simple environment does not stress the virtual reality hardware, containing a low number of polygons to render, but provides context for significant elevation changes in a small area. Within the environment, participants had access to a single total station and tripod. Two prism rods were available. This matches a realistic surveying scenario, where one person would operate the tripod, while others would hold prism rods at designated locations.

2.5.2 Measures

As the study was mostly about obtaining user feedback, surveys were the primary instruments, each administered electronically using a Google Form at the experiment facility. A background survey gathered variables of interest, such as age and gender, vision correction glasses use, experience with VR, gaming, and surveying, and for dyads, how well they knew their partner. Three common post-experience surveys were also used. The NASA Task Load Index (TLX) assesses task difficulty along several dimensions [46]. The Steed-Usoh-

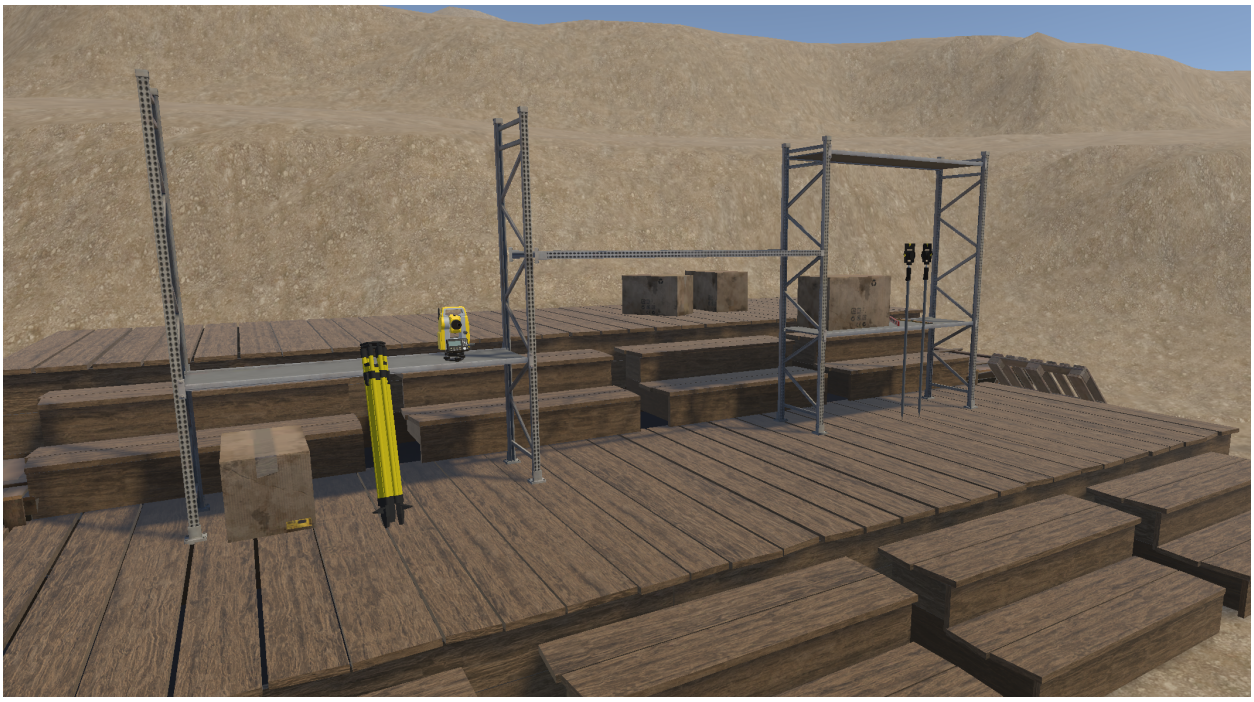

Figure 2.8: The virtual environment used for the pilot study. One total station and tripod are available, with two prism rods.

Slater presence questionnaire [114] and Bailenson co-presence questionnaire [4] measure their eponymous constructs. Lastly, a feedback survey was administered that asked students to rate their opinion on usefulness of the tool for practice, their perceived likelihood to use it independently for study, and their perceived performance in the task. Finally, there were two questions for open-ended feedback on what difficulties they experienced and for additional comments about the experience.

In addition to survey measures, all writing done using the whiteboards was logged, as were a detailed log of all participant movements (head and hands) and actions taken (e.g. buttons pressed).

2.5.3 Procedure and Task

The majority of participants signed up and arrived in groups of two, however a small portion of the participants arrived alone. This formed three study groups, labelled as GROUP (N=48, 24 dyads) - corresponding to dyads who both trained and were evaluated as a group, INDIVIDUAL (N=33, 15 complete dyads, 3 single datapoint due to elimination of underage participant) - corresponding to dyads who trained as a group, but were evaluated individually, and ALONE (N=4) - corresponding to those who were trained individually and evaluated individually. In all cases, the experimenter training the participants was the same (1st author). In the INDIVIDUAL case, a second experimenter (2nd author) was involved in the evaluation.

After completing a brief background and demographics survey, the experimenter explained how to wear the headset and hold the controllers. Two separate, non-overlapping areas were designated for each participant in the same room, so as to avoid physical collisions. Once both participants were wearing the headsets, the experimenter continued the instruction of the system from within VR using a third VR system. Participants were walked through the teleportation system, entering their name and choosing a color so that they could more easily identify each other. Following this, they were trained on using the virtual surveying equipment, first in a staging environment (called the "hub" world), and then to perform total station setup and take measurements in the quarry scene. Afterwards, each study group performed the setup and measurement without experimenter assistance (unless requested). For the INDIVIDUAL group, they were taken to separate (but identical) quarry scenes, where

the second experimenter was involved in case help was needed. After completing their task, they filled out the post-experience surveys and were sent the results via email so that they could write their report for the class assignments.

2.5.4 Results

Background Survey

Results from the background survey were consistent with recruitment from a freshman-level undergraduate introduction to engineering class. A majority of the 85 participants were male (24 Female, 61 Male) with an average age of 19.1 years. Considering dyads, a majority were same gender (32 out of 39), likely the result of coordination in the sign-up process. As seen in Fig. 2.9, surveying and VR experience were low for both genders, with slightly more experience indicated by male participants (M=1.97,SD=1.18) than female participants (M=1.46,SD=0.66). Game playing experience was significantly ($p < .05$) higher for male participants (M=3.38, SD=1.98) than female participants (M=1.92, SD=1.18). Male participants indicated a slightly higher rating of knowing their partners (M=3.70,SD=1.80) than female participants (M=3.09,SD=1.31).

Presence and Copresence

The preferred technique was used to calculate presence as the number of answers on the questionnaire of 6 or 7 (indicating a high level of agreement or above, maximum 7). Despite controlling for game playing experience, it was found that presence was significantly higher ($p = 0.018$) for male participants (M=2.57,SD=1.77) than female participants (M=2.00,SD=1.56).

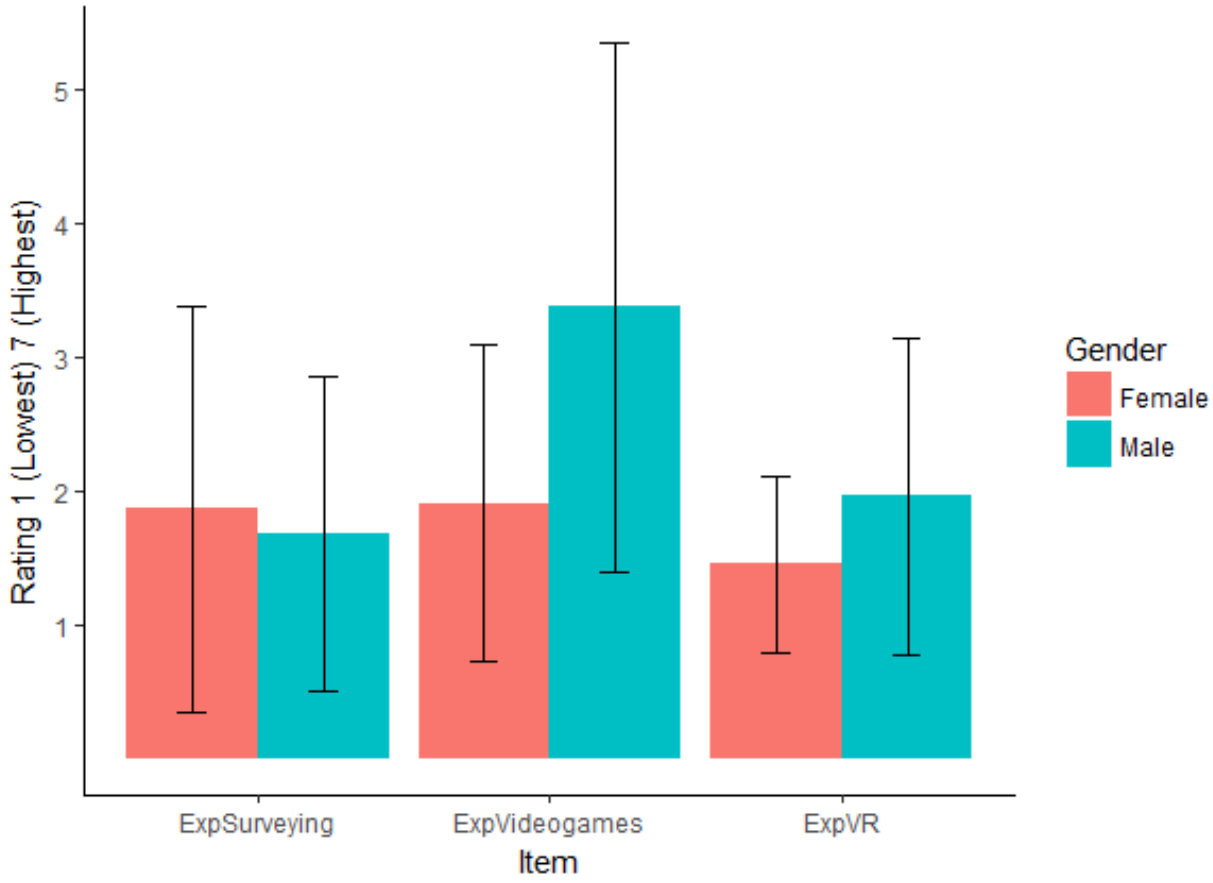

Figure 2.9: Background Survey Results

For copresence, the same technique was used (maximum 3). As expected, those in the GROUP condition rated copresence significantly ($p < .001$) higher (M=2.3, SD=.99) than the INDIVIDUAL condition (M=1.55, SD=1.15). The SINGLE condition (N=4) was surprisingly highest for this category (M=2.5, SD=.58), though the sample size was small, and acknowledging the fact that in all cases, the users were in the virtual reality world with at least one experimenter.

Usability and Difficulty

No significant differences were found for gender or study condition for any of the items on the usefulness survey. However, these ratings (shown in Fig. 2.10 were quite high, suggesting strong student enthusiasm for the system. Participants generally thought the system was easy to use (M=5.80,SD=1.01), and they thought it would be useful as a practice (M=6.35, SD=1.02) and study (M=5.74, SD=1.44) tool. However, as shown in 2.11, the NASA TLX instrument showed signs of higher task load along the mental ($p < .01$), temporal ($p = .07$), and frustration ($p < .05$) dimensions for female participants.

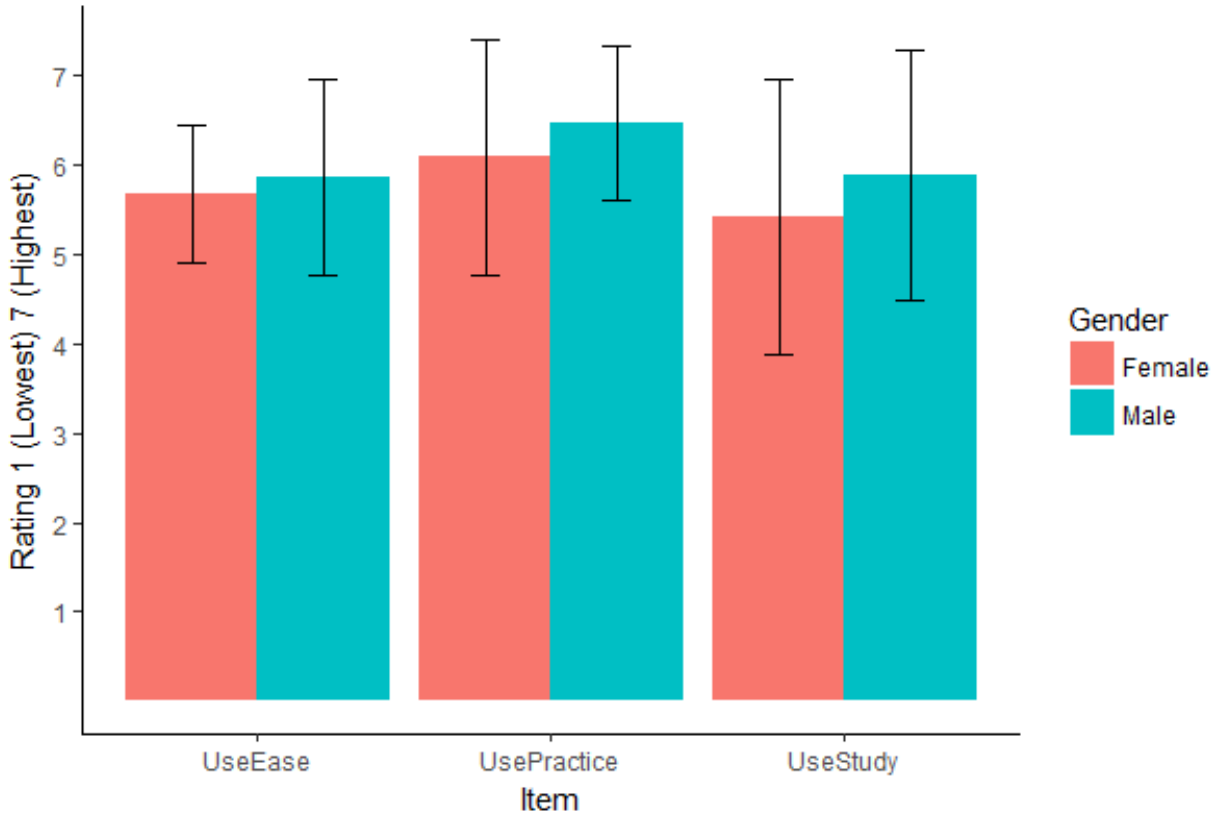

Figure 2.10: Usefulness Survey Results

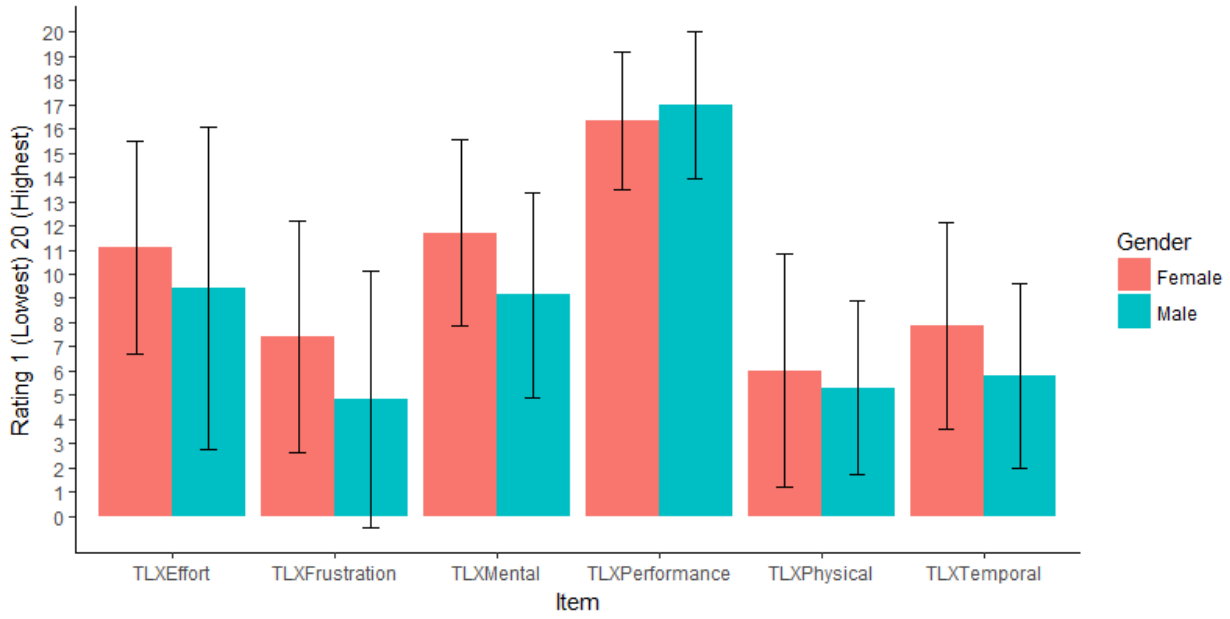

Figure 2.11: NASA TLX Results

Comments on usability were also analyzed by determining the amount of overlap in user issues reported about various system components. Of this, the most common issue reported was a difficulty in reading text (reported by 24 participants). As discussed in Section 2.4, text legibility is directly related to the size of the text and distance to the headset. Lack of user experience with virtual reality, coupled with small text used on the total station, likely contributed to this issue. The next most common issue reported was with movement (reported by 12 participants). Again, this is likely related to lack of user experience, and overall, quite a low incidence relative to the amount of experience with virtual reality). Encouragingly, leveling the total station was also reported as an issue by 12 students. This was intentionally the hardest part, and so it was seen as a validation measure of the task. Other notable issues

included writing (11) and turning objects/knobs (9). Both of these issues are due to the lack of tactile feedback in virtual reality, and are likely also related to lack of virtual reality experience.

2.6 Conclusions, Limitations, and Future Work

The current work evaluated a novel, immersive virtual reality application for engineering education in the context of land surveying training. Results from a study of 85 students in a freshman-level introduction to engineering course suggest both high usability and student enthusiasm for the approach. A concern remains, however, that there will be gender disparities related to the use of virtual reality systems. The current study showed small, but significant biases favoring male participants on several key (self reported) factors including task load and presence, even after controlling for video game playing experience. Further studies are necessary to determine if these effects would persist with consistent exposure and training in virtual reality and in objective measures such as task performance, which was difficult to measure with the current group.

Related to this, a clear limitation of the current work is that it was not used to teach students as part of a land surveying course, and thus lacks ecological validity. However, this was necessary, as integrating into a course would require more infrastructure. While the cost is not a major issue relative to the cost of surveying equipment, there are still logistical issues to deal with in terms of user training and improvements in usability before course integration becomes viable. A pilot study on course integration is underway, which aims to

determine the value that experienced surveying students perceive, and the extent to which their prior experience allows them to perform virtual surveys on the mimicked equipment. This is also being performed with a more authentic task that asks users to determine the volume of material that has eroded from a hill into a nearby river.

The ENGREDUVR platform is also evolving into a course-ready tool. Key improvements have been made that allow students to go through an automated tutorial, and to form *ad hoc* groups by entering a "group code" that can be shared through ordinary communication channels. This will enable a virtual reality laboratory to run without much human assistance, or for students to use their own virtual reality devices. As this happens, more opportunities will be found to leverage the equipment throughout a curriculum, with potentially compounding benefits, both logistical and educational.

Chapter 3

Virtual Reality Point Cloud Annotation[1]

[1]Franzluebbers, Anton and Li, Changying and Paterson, Andrew and Johnsen, Kyle. 2022. SUI '23. Reprinted here with permission of the publisher.

Figure 3.1: LIDAR scanned cotton plants visualized as point clouds in various states of annotation used for the user study. The two young cotton plants on the left were used for an annotation task, and the two mature plants on the right were used for a cotton boll counting task.

3.1 Abstract

This work presents a hybrid immersive headset- and desktop-based virtual reality (VR) visualization and annotation system for point clouds, oriented towards application on laser scans of plants. The system can be used to paint regions or individual points with fine detail, while using compute shaders to address performance limitations when working with large, dense point clouds. The system can either be used with an immersive VR headset and tracked controllers, or with mouse and keyboard on a 2D monitor using the same underlying rendering systems. A within-subjects user study (N=16) was conducted to compare these interfaces for annotation and counting tasks. Results showed a strong user preference for the immersive virtual reality interface, likely as a result of perceived and actual significant

differences in task performance. This was especially true for annotation tasks, where users could rapidly identify, reach and paint over target regions, reaching high levels of accuracy with minimal time, but we found nuances in the ways users approached the tasks in the two systems.

3.2 Introduction

Many important real-world applications obtain and process digital representations of built and natural environments. These surveying activities are vital for scientific and engineering pursuits, as they provide a means to measure physical traits of the environments and how they change over time. Commonly, scans of the environment are created using methods such as light detection and ranging (LIDAR) or photogrammetry, which output discrete spatial samples, known as point clouds. From here, a variety of automated and manual tasks ensue. Tasks involving human interaction with these point clouds, such as counting, measuring, and annotation, may be well-suited for immersive virtual reality (VR), which, through stereoscopic, head-tracked rendering, could provide a better sense of measurement depth and also improve navigation performance. Further, tracked controllers commonly used in VR may improve selection of point data. Though immersive LIDAR visualization tools exist (e.g. LIDAR Viewer [56]), there is little evidence that they improve task performance over desktop-based tools (e.g. Cloud Compare or Mesh Lab). To both address and better understand this gap, we aimed to design a VR-based tool. This paper reports on that

tool, including a study that addresses the critical question of how VR may improve task performance relative to desktop tools.

Beyond visualization, one of the contemporary use cases for such a tool is to provide labelled training data for automatic semantic segmentation, which allows higher level information processing. Manual annotation is required, as can be seen in public datasets with pre-annotated features, such as the DublinCity [139], Semanticposs [89], and Semantic3D [43] datasets. These are often quite large in scale, and are generally useful for self-driving vehicle applications, as this is one of the foremost uses of raw point cloud data for the present and near future. As machine learning methods become more widespread, this technique is being applied to a wider variety of fields, and the development of these new algorithms requires new ground-truth datasets, which often do not exist for application-specific annotation, counting, or classification problems. The intent of this work is not to contribute to this body of datasets directly, but to present and analyze a system that makes the development of new datasets more accurate and efficient, especially for applications that require datasets that do not exist, since their focus is too narrow or too innovative for an existing effort to have occurred.

This work was motivated by one such application, a research project aiming to automate and accelerate the measurement of agricultural products. As a part of this project, LIDAR point cloud datasets of cotton plants were produced, and then subsequently processed and annotated, either directly using a point cloud visualization toolkit, or more recently, by automatic annotation algorithms. The development and training of these algorithms via machine learning depends heavily on a large variety of point cloud data that are annotated correctly on a per-point level, or "ground-truth," through which their effectiveness could

be evaluated. Given that this was seen as a tedious task using existing tools, we aimed to develop a tailored tool optimized for the process, and were particularly interested in the value of headset VR interfaces.

VR displays have immediate benefits in depth perception over traditional 2D displays through both stereoscopic display technology as well as parallax shift from motion tracking [129, 42]. The interaction affordances of scenes viewed in this manner may also change, especially when combined with 6-Degree-of-Freedom (DoF) motion controllers. For example, users may reach into the point cloud, directly selecting points, or move about the environment differently, such as by walking and turning in the real world. Enabling these actions with dense, natural point clouds is a major performance challenge. VR experiences must run at high frame rates to remain usable and to reduce simulator sickness, but incur significant rendering performance overhead due to high resolution stereoscopic rendering. To mitigate this, our application uses a continuous level of detail (LOD) system, with the novel variation of applying LOD emphasis on the hand positions, facilitating annotation work. We also addressed performance issues encountered when selecting points, and by doing this, enabled fluid, high frame rate point cloud interactions in immersive VR.

In addition, we aimed to quantify the task performance gains with this system, especially compared to a desktop interface. Thus, we also co-designed such an interface that could be used for the same tasks, but was optimized for mouse-keyboard-monitor interactions. These interfaces were compared through a within-subjects user study (N=16), where participants performed annotation and counting tasks on LIDAR scans of cotton plants and provided feedback on usability. Large performance advantages were found for the VR interface on the

point annotation task, especially during early phases, where precision may be less important, but this advantage became much smaller as the task neared completion. Counting performance was similar, with a small, but significant advantage for the VR interface. Surveys and written responses qualify these results, highlighting that users strongly preferred the VR interface for both tasks, mostly due to a perceived performance advantage (even though this was not always the case). Altogether, results were promising that our VR interface could accelerate, and make more enjoyable, the currently tedious tasks on point cloud datasets, especially given that there is significant room for improvement in VR hardware and interface design.

3.3 Related Works

3.3.1 Immersive LIDAR Visualization

Immersive data visualization became practical in the early 1990s with the introduction of the CAVE [22], a cube of projection displays that surround the user. CAVEs, and similar spatially immersive displays, have since been used for a variety of visualization applications in architecture, astronomy, mathematics, biology, medicine, and physics. [23, 75]. Kreylos et al. was one of the first to explore large-scale immersive point cloud visualization and interaction, resulting in the open source LIDAR Viewer application [56], which has been used extensively for geographic science visualizations [58, 107]. This system was originally designed to be used with CAVEs, which provide access to high-performance computing and support multiple users [51]. Though visualization walls have remained popular [36], recent

efforts have brought LIDAR Viewer and other similar systems to personal computers and personal VR headsets [55, 72, 102, 57, 135], greatly lowering the barrier for use.

Performance has been a dominant theme in the above LIDAR visualization research. LIDAR point clouds tend to be large, greatly exceeding the boundaries of system memory, requiring so-called "out-of-core" techniques that pre-process and subdivide datasets into hierarchical levels-of-detail (LOD), loading increased LOD into memory as the camera approaches. The exceptional performance of this technique is highlighted by web-based point cloud viewers, such as Potree [102] that gracefully reduce performance to run on most hardware. However, Schütz et al. point out that it is worthwhile to explore the maximization of in-memory point cloud visualization, i.e. the maximum number of points that can be precisely visualized in real time. They presented a continuous LOD method that they classify as a "view-dependent, continuous LOD method with fidelity-based simplifications [101]." Their system creates a lower resolution version of the full point cloud based on both gaze direction and viewpoint position and is updated regularly to account for movements of the camera using a compute shader. This method was found to effectively render point clouds of up to 104M points with a GTX 1080 on an HTC VIVE Pro. The continuous LOD system was found to have more "subtle and less irritating changes of detail as users move through the scene" compared to discrete LOD methods. Another, related approach furthers the idea of fully in-memory rendering, dividing work to render the cloud over multiple frames [111, 103].

Commercial software solutions also exist for the purposes of visualizing large points clouds both on desktop displays and in VR, such as Nubigon [2] and Interviews3D [3]. Our work builds upon existing efforts, with a focus on agricultural LIDAR datasets that have vast differences in point density, owing to the need to capture fine features of plants by capturing data from multiple perspectives. We focus on in-memory visualization, as our tasks are isolated to single plants at a time, and hence levels of detail offer few advantages.

3.3.2 Annotation Interfaces

Annotation is the process of ascribing data beyond that contained in the original data set, and is a common goal for machine learning algorithms to complete automatically. For scanned environments, this is often assigning each point or region a label (e.g. chair, teapot). Algorithms for this tend to rely upon supervised machine learning techniques, which in turn rely upon manually annotated data as a ground-truth for training and evaluation. For example, there are several large pre-labeled points that are available to use for this kind of training, such as the TUBS Road User Dataset with 12,000 labeled scans [95], the H3D dataset with 1 million labeled objects from 27,000 scans [90], or the semantic3D.net dataset with over 4 billion points, which was created as a benchmark for other machine learning classification algorithms [43]. These datasets were each annotated using either manual or semi-automatic methods, making annotation a key, though tedious, part of the pipeline, and the interface to this has received attention from the 3DUI research community.

[2] https://www.nubigon.com/
[3] https://www.3dinteractive.de/products/interviews3d/

Wirth et al. investigated whether the advanced 3D visualization and input capabilities of an immersive VR headset would improve annotation performance [132]. Points were selected by placing and scaling boxes using the position and rotation of the 6-DoF Oculus Rift Touch controllers. The authors then evaluated the performance of the technique using a method detailed in Monica et al. [79], basing the evaluation on both time to completion and accuracy of the annotation result. The work of Monica et al. made use of manually placed control points in the middle of segment regions, and then calculated the correct segment for all the points in the cloud based on "distance" using a cost function based on position, normal, and color of the points. Wirth et al.'s VR method compared favorably in all metrics to this method of control point based selection.

Li et al. developed a similar point cloud annotation software called SUSTech POINTS [64]. The software is designed as a web application to be used on a computer monitor. The main interface is divided into several sub-views showing top, side, front, and perspective views, along with a context photo of the environment from which the point cloud was scanned. 3D annotation is performed by creating and transforming boxes around the selected points. Once a box is placed, the user has options for adjusting its 3D transform using gizmos. The box size can be adjusted automatically by scaling to tightly enclose the points it contains. Since the tool is designed for use on LIDAR data streams, which contain consecutive frames of similar data, an algorithm enables the transfer of annotations between frames. This allows the user to then manually adjust the annotation to fit the new data. Annotation efficiency was measured using both accuracy and speed as metrics.

Several other works have investigated the use of alternate input methods to improve the point cloud annotation or manipulation process. BioVR uses the Leap Motion hand tracking system to eliminate the need for tracked controllers while visualizing and manipulating similar bio-informatics datasets [135]. Bacim et al. designed a system to annotate point clouds based on the concept of repeated bisection of the data [3]. Due to the reduced input complexity of bisection, they were able to use a Leap Motion to perform the slicing. In order to change the viewpoint of the camera, a six-degree-of-freedom 3DConnexion SpacePilot Pro 3D mouse was used. The authors noted the use of progressive refinement was critical to allowing a less precise input device such as the Leap Motion to still provide sufficient annotation accuracy. Veit et al. proposed a point cloud annotation framework that makes use of a tracked smartphone for the selection of points [116]. The touchscreen makes text entry for the labels convenient. HyFinBall was developed to meld the advantages of traditional 2D inputs and 3D natural user interfaces by using multi-touch ball input devices that can be used while resting on a surface [18]. The authors tested their device by exploring LIDAR datasets.

It has been found that VR offers advantages over 2D modalities for complex 3D visualization tasks, and our work confirms these findings using using modern hardware and a new applied task [11, 10]. Surprisingly, few works have evaluated differences in task performance between different user interfaces for this type of application [61]. As such, alongside our designs for the immersive and non-immersive interfaces, we also provide evidence of how the immersive interface improves certain aspects of the annotation process.

3.4 System Design

As mentioned in Section 1, our system was designed for visualizing point cloud datasets acquired through LIDAR scanning of cotton plants, with the primary task of annotating point clouds at the individual point level (as opposed to bounding regions). The raw data were obtained with a Faro Scanner Focus 3D, a rotating LIDAR system that provides colored point clouds. As multiple scans had to be taken to account for occlusion, these were first merged into a single, high density, point cloud (see Figure 4.1) using the CloudCompare software. The data were then exported into the Point Cloud Library's point cloud data (PCD) file format, which is an emerging standard and offers significant interoperability between point cloud tools. From here, they can be processed by our application.

Our application[4] was designed entirely within the Unity3D game engine (version 2020.3 LTS), which supports a wide range of operating systems and hardware. However, it does not natively load PCD files, which required a custom importer and exporter to be built. To minimize time consuming steps in the pipeline between CloudCompare and Unity, PCD files are parsed and loaded directly into memory and then briefly processed to determine their extents such that their size can be automatically normalized. We use a compute shader for this processing, which takes less than 10 ms for point clouds of several million points on an NVidia Titan V graphics processing unity (GPU). All data is loaded into a compute buffer on the GPU, with 20 bytes stored per point; 12 bytes are used by the 3 single-precision floating point numbers for the xyz position of the point, and the remaining bytes are used to

[4]Source is available at https://github.com/velaboratory/DataFoldvr-Virtual-Reality-Point-Cloud-Annotation

store color, assigned layer, and a reference, or "correct" layer, which was only used for the user study to give immediate feedback on completion percentage. Without this last reference value, the data could be packed into 16 bytes.

3.4.1 Point Cloud Rendering and Annotating

Immediately upon loading our first point clouds, which are extremely dense relative to their real-world scale (a cotton plant is roughly a 1-meter cube, with scans being in some cases over 7 million points), we found significant performance issues when attempting to visualize them in VR at interactive frame rates. Modern GPUs are capable of processing and rendering millions, even hundreds of millions, of triangles per frame at interactive rates. However, doing so while all triangles are visible on screen (i.e., they are not culled and are shaded), with significant overdraw (triangles are not sorted), and while processing them for annotation tasks, severely limits performance. This is compounded by the high resolution and frame rate requirements of VR systems. These problems do not typically occur with lower-density clouds such as environmental scans, as an LOD system can automatically reduce the number of in-memory points being rendered based on camera distance. Doing so when all points are reachable and clearly visible introduces quality issues, as details pop in and out.

In order to maintain high visual quality and detail while remaining performant with the high requirements of a VR-enabled application, our system implemented a continuous LOD similar to that presented in the work by Schütz et al. [101]. However one additional feature was implemented to improve the utility of the downsampling technique when dealing with precise input. While the original work changed density based on view direction and head

Figure 3.2: An example of the continuous LOD system used to render the point clouds. This system ensures that the full detail present in the source data is always visible in the area near the cursor/tracked hands, which happens to be in the center of the image here. The points in this image are reduced in size to exaggerate the low density areas and emphasize the effect.

gaze, our system uses the hand positions to smoothly increase the point cloud render density in the target area. This allowed the area of focus to be guaranteed to have the maximum density, while limiting the total number of points to a modest point budget. To achieve this, all of the points are added to a compute buffer. A compute shader is dispatched on the GPU every frame to select the appropriate subset of points to add to an AppendBuffer. This output buffer is then rendered using Unity's DrawProceduralIndirectNow method, avoiding the expensive operation of transferring data between the CPU and GPU memory. From here, the number of triangles rendered was variable (between 2 and 12), as produced by a geometry shader, ranging from a circular shape to a quad, depending on distance from the camera, based on Keijiro Takahashi's PCX example (https://github.com/keijiro/Pcx). The effect can be seen in Figure 3.2.

Beyond visualization, performance was a major concern when developing realtime annotation, and again compute shaders were used heavily to speed up the process. CPU-based implementations may use indexed data structures to retrieve close points to the cursor, and subsequently annotate them. However, this data is also needed on the GPU for visualization, and in our case, the high density of the point clouds would result in large transfers between the CPU and GPU. Instead, we inverted the problem for GPU implementation. For every frame that annotation is indicated over a designated region, a compute shader performs a distance calculation on the GPU for every point. If the point is within the region, the byte corresponding to the current layer index is changed to match the desired layer. During the rendering process, this layer index is used by the fragment shader to draw the points in either

original color, a distinct solid color, or to hide the point. This allows the operation to proceed in parallel entirely on the GPU, greatly accelerating the process.

3.4.2 User Interfaces

Two interfaces were designed for the study, which we abbreviate as our "*VR*" and "*2D*" interface going forward. The *VR* interface was designed for use with a stereoscopic VR headset and two 6-DoF controllers. The *2D* interface was designed for a single-monitor, keyboard and mouse setup. These interfaces could be swapped in real time, via a keyboard switch or by clicking on a button. When in *VR* mode, the monitor would show a mirror of the VR-view. Both interfaces could be in "counting" or "annotation" modes, depending on the task, which slightly varied the interface to show relevant feedback to users. In the study, these were switched by the experimenter, but would otherwise be switchable by the user, meaning that both *2D* and *VR* could be used in practice, though not simultaneously in the current implementation.

The *VR* condition interface is depicted in Figure 3.3. The user was spawned into a virtual room large enough to easily accommodate the point cloud, which was placed in the center according to its bounds. A virtual tablet was used to show task-relevant UI elements, which was most prominent in the counting task. All controls were identical on both controllers, eliminating the need for dominant hand preferences and simplifying the learning experience. The virtual tablet is called up by users by pressing the B or Y button on the Oculus Touch controller, allowing them to use the opposite controller to make selections (depicted by a 2D

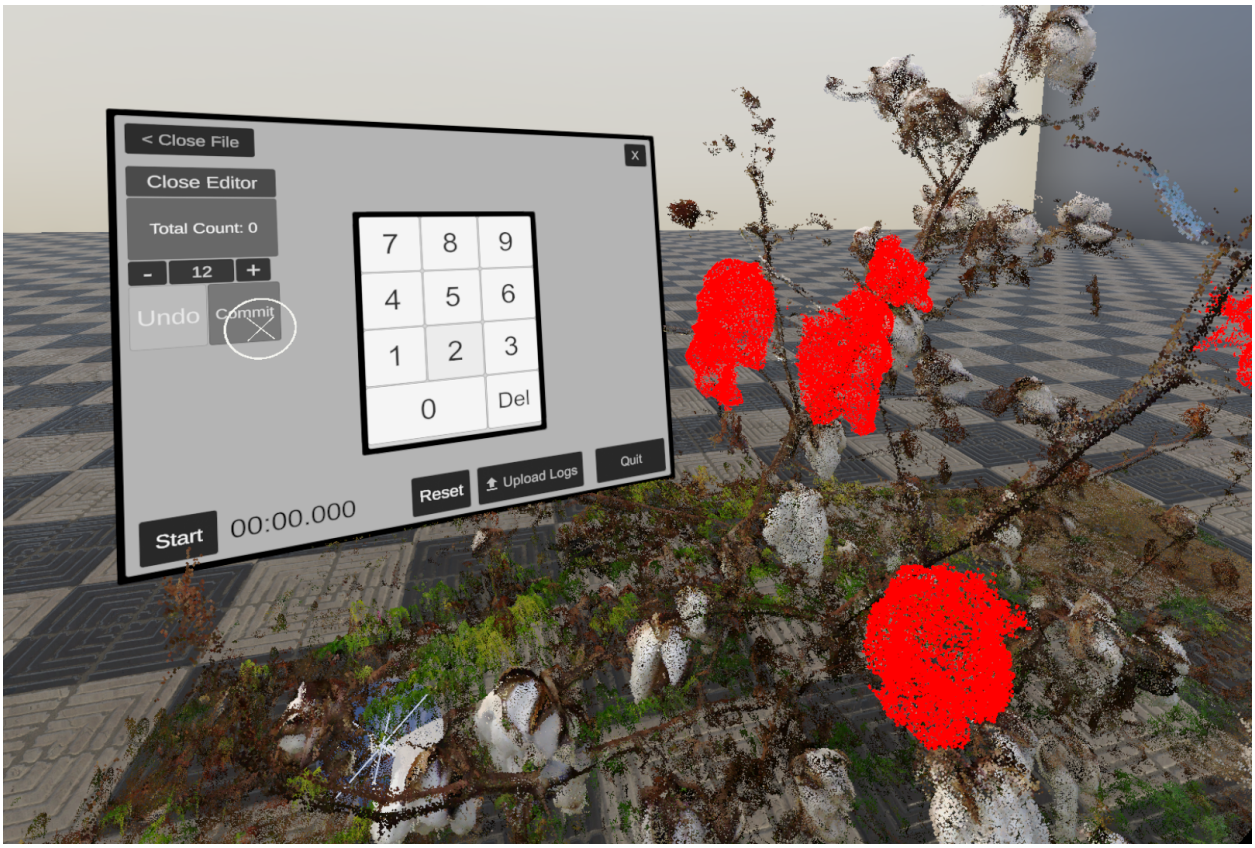

Figure 3.3: Screenshot of the counting task in the *VR* condition. The tablet interface is visible, on which participants entered their subtotals when counting groups of bolls.

cursor that appeared on the tablet when the controller was held near it). In the annotation task, the completion percentage was shown near the lower part of the controller.

In order to assign points to a particular label, a painting metaphor was used. In the *VR* condition, this was done by visualizing a sphere around each of the users' hands, and allowing them to mark all the points within the sphere for every frame that the index trigger on that controller was held. To show the marked points, the color of the points was changed from the original scan color to a solid color. The user was able to select a layer using a color

picker, though in the user study only a single color was used, and a secondary controller button (A or X) was used to erase previously painted points and return them to the original color. The spheres could be scaled by moving either controller thumbstick on its Y axis to allow for more broad or precise painting actions.

While the point clouds were relatively small in real-world scale, and thus walkable within a small space, a VR locomotion system was added to make the system usable in a seated position, and the study was performed seated to match the desktop interface. Rather than transforming the point cloud, as other systems have done [56], we chose to keep the transform between the virtual room and point cloud fixed and instead moved the user's position with a combination of pulling motions (either hand) and use of the thumbstick for rotations. When the hand grip trigger was held down, the world was pulled to match the controller position, with a small amount of momentum if the grip was released while moving. Smooth rotation (at a relatively high rate of 200 degrees/second) was activated by the X axis on the thumbstick. When combining smooth rotation and world grabbing, the origin of the rotation was moved to the hand position. In our study, participants did not show any significant difficulty getting used to the controls, and more experienced VR users were able to use this system to paint with one hand and move with the other hand simultaneously in one smooth motion. In addition, we did not have any reports of nausea or discomfort from users.

The *2D* condition interface is shown in Figure 3.4. In contrast to the *VR* condition interface, the user is shown only the point cloud, rather than a virtual room, as their environment is the real room. The on-screen interface, though custom designed, mimics the features found in other annotation systems, including the ability to choose from a variety

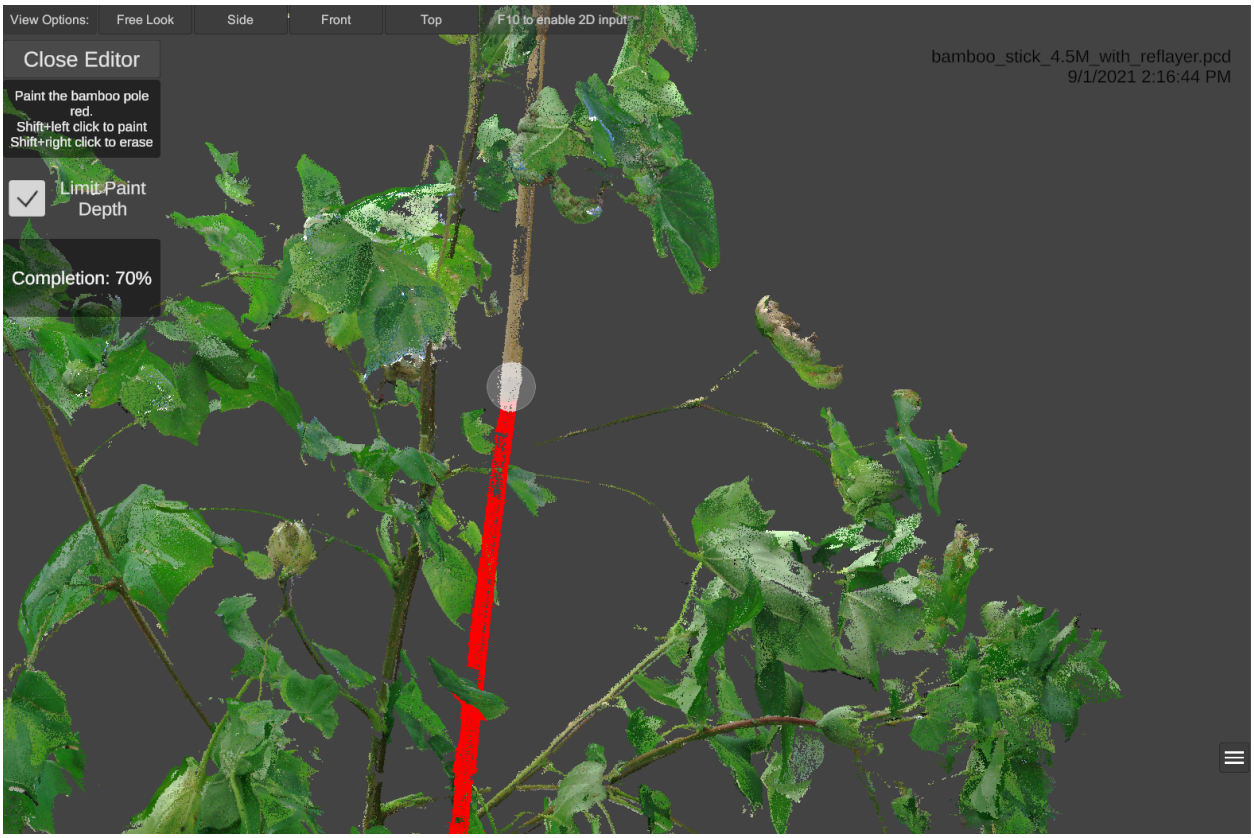

Figure 3.4: Screenshot of the annotation task in the *2D* condition. The semi-transparent circular cursor can be adjusted in size.

of aligned orthographic views of the cloud, or to "free look" with the mouse. The mouse operated in two modes, one for navigation and the other for annotation. In navigation mode, the left mouse button was used to select, middle to pan the orbit point in view space, scroll-wheel to zoom, and right to orbit around that point. In annotation mode, which was accessed by holding the shift key, the left mouse button would annotate selected points, the right would remove the annotation, and the scroll-wheel would change the size of the selection region.

On a traditional computer display with mouse interaction, the selection of a specific sphere of influence is more difficult. Initially, the system was designed to paint all points overlayed by the 2D circular cursor, essentially creating an infinitely long cylinder of influence. While this difference between the *2D* and *VR* conditions could be considered fair because it arose as a result of a fundamental difference between the two systems, an additional feature was added to the *2D* condition to achieve a more similar spherical influence effect. To achieve this, a compute shader calculated and returned the points within a smaller circle 1/5 the size of the current cursor size to the CPU. The resulting points were then iterated on the CPU to determine the point closest to the camera. This point was then used by the annotation shader such that the depth of selection was limited to the diameter of the circular region (essentially placing a depth-limited cylinder at the cursor point). This approach prevented background points from being selected accidentally, as seen in Figure 3.5. We also included a toggle button for this behavior, as the infinite cylinder could also be useful, though the majority of participants left the feature on (the default). Before pressing the button that performed the paint operation, the points were highlighted by increasing the brightness of the original color, as seen in Figure 3.6. This was especially useful for the *2D* condition with the depth limit feature enabled, since not all points under the cursor would be painted, though the highlight effect was visible in both conditions.

Figure 3.5: An example of the depth limiting algorithm used in the *2D* system. The top images show the initial painted view. The bottom two images show the same cotton boll from a rotated perspective after the paint operation. The left two images have the depth limit turned off, and the right two have the feature turned on.

Figure 3.6: The highlight effect previewing the points that will be selected by a potential paint operation in the *2D* interface. In this image, the depth limit feature is preventing the background points from being selected. The *VR* condition also has this highlight effect for points within the hand spheres.

3.5 Study design

Our study was designed to evaluate and compare the two interfaces for two tasks that are commonly performed on point clouds: precise annotation and feature counting. A within-subjects, crossover design was used, such that each participant used each interface once for each type of task, resulting in a sequence of four tasks. The order of *VR-2D* was alternated in order to account for the learning effect, so that half of the participants used order *VR-2D-2D-VR* and half used *2D-VR-VR-2D*, with the first two instances being an annotation task, and the last two being the counting task. The order of the annotation and counting tasks was fixed, as there was no intention to compare between the two tasks and point clouds. In addition, we decided to use the same point cloud for completing the *VR* and *2D* versions of each task for every participant, as it would be difficult to find different point clouds with similar difficulty, avoiding a source of bias or variation in the timing between *VR* and *2D*. Each task consisted of a training phase and a testing phase, as described below.

3.5.1 Tasks

In the **annotation task**, participants were instructed to precisely paint a portion of the point cloud according to the part of the object that it represents. In the training phase, the point cloud was a scan of a relatively small cotton plant in a pot. The task was to mark the points that belonged to the plant itself, excluding the pot, soil, and ground. The evaluation phase was a larger potted cotton plant with a bamboo pole used as a support. The task was to mark the points belonging only to the bamboo pole, which was colored significantly

lighter than the rest of the plant or the plant's stem. As the plant was growing around the pole in all directions, this task was significantly more difficult, and impossible to perform from a single perspective, though the pole was nearly straight. The task was completed by achieving a 98 percent score. The score was determined by comparing the present annotation to a ground-truth annotation. As the ground-truth was also approximate, 98 percent was chosen empirically as being consistently achievable. The percentage indicated was the F_1 score (see Equations 3.1-3.3), which requires eliminating false positives (annotating incorrect points) and false negatives (not annotating the correct points), such that the user could not simply select the entire point cloud to achieve a high score.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{3.1}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3.2}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.3}$$

In the **counting task**, participants were asked to count cotton bolls that were scattered throughout the plant. This task would be challenging to perform in the field without picking the cotton. The training and testing tasks were similar, except that the testing task was a larger plant with more bolls. During the task, participants could mark already counted bolls with the painting interface. In addition, participants were given an interface to serve as a counter, and if they desired, to "commit" that count, which would erase the currently painted

areas akin to picking the cotton. The task was completed when the participant indicated that they had counted all bolls.

3.5.2 Hypotheses

The following hypotheses were formulated based on prior work and pilot testing, which suggested that the *VR* interface would enable superior task performance and result in higher levels of user satisfaction.

1. **Participants will be faster overall with the *VR* system in the counting task.** We expected *VR* to allow for more rapid identification of bolls, quicker marking of counted bolls, and easier navigation, resulting in faster performance.

2. **Boll counts will be higher and more consistent when counting with the *VR* system.** The effects of stereoscopic rendering and smoother parallax motion present in the *VR* condition will result in fewer missed bolls due to occlusion and a more precise count.

3. **Participants will report higher accuracy and perceived efficiency using the *VR* system.** Similar to the previous hypothesis, the *VR* condition will present information in a more intuitive format that will be perceived as better for these tasks.

4. **Participants will prefer to use the *VR* system.** Influences such as the novelty effect and the more intuitive nature of the *VR* system will overwhelm any negative effects such as comfort caused by wearing the headset.

3.5.3 Study Population and Environment

Following human-subjects review approval, 16 participants were recruited from the local university population and lab groups to participate in the study by direct invitation and word-of-mouth. The study took place over the course of 3 weeks, and was performed within our research laboratory. For the *VR* condition, an Oculus Quest 2 headset (1832x1920 resolution) using the wired Oculus Link feature set to 90Hz and default 2448×2688 per eye rendering resolution was connected to a PC with a i9-7980XE CPU and an NVidia Titan V GPU. As described earlier, the application frame rate was verified to have remained at a constant 90 frames per second throughout the study, and the Oculus Link cable data rate was set to the maximum of 500 Mbps to avoid visible compression artifacts. Our study used a point budget of 3 million for point clouds of up to 7.8 million points, as seen in Table 3.1, though none of the participants remarked that they noticed the LOD system at all. This budget was empirically chosen because it offered sufficient visual quality for the selected points and fell well within the testing computer's performance limits, even while recording the screen, rendering to multiple monitors, and using the Oculus Link encoder. For the *2D* condition, a standard mouse and mechanical keyboard were used with a 32 inch 2560x1440 monitor at 60 Hz. We chose to retain the 3 million point budget for this condition to be consistent with the possible information available within the *VR* condition at any given time, although no LOD optimization was strictly necessary to maintain 60 frames per second.

Table 3.1: Point cloud sizes used in the experiment

Task	Point Cloud Size
Annotation Training	3,844,639
Annotation	4,485,971
Counting Training	2,609,825
Counting	7,867,195

3.5.4 Procedure and Measures

After getting informed consent, participants filled out a background survey with questions about demographics and prior experience with VR and point cloud data. As described earlier, the study consisted of two task-types, annotation and counting, which were repeated in the *VR* and *2D* conditions. Each task consisted of a training session as well as a timed session on a larger and more complex point cloud. During the training session, the participant was instructed by the researcher on how to perform the task and how to use the controls for both viewing and interacting with the point cloud. While the training task was timed, the participants were encouraged to take their time and get used to the controls. Once the participant completed the task on the training point cloud, they were moved on to the larger point cloud of the same task. They were instructed to complete the task as quickly as possible and were given minimal external instruction during the activity. After completing the task in VR, participants performed the exact same task using the same point clouds using the *2D* interface before switching to the second task in the opposite order. During both the *2D* and *VR* training periods, time was only logged while the participant, and not the researcher, was actively operating the controls themselves. We excluded the time when the researcher

was demonstrating the controls. In both conditions, the participant was only moved on to the main task after completing the task and reporting that they were comfortable with the controls and task. Brief questionnaires were given between every task, asking for any sources of difficulties after each of the four tasks and relative preference scores between the *VR* and *2D* cases after both the annotating and counting groups of tasks.

3.6 Results

All participants were able to complete all tasks successfully, though one participant was asked to restart the first 30 seconds of one of the tasks due to a software crash. The average age was 28 yrs (SD=9.26). Of the 16 participants, 12 reported male, 3 reported female, and 1 declined to identify. There was a mix of very experienced and inexperienced VR users, but the majority (9/16) were very comfortable with 3D video games. Completion of the study took between 30 minutes and 1 hour for all participants.

Using a Chi-square test for independence, no significant differences were found between the two groups assigned to different condition orders. In addition, none of these background variables (shown in Figure 3.7) correlated strongly with the performance variables, so they were not used as covariables in subsequent analyses. No motion sickness, vision problems, or discomfort were reported during the study.

Three relative preference questions were asked for each of the two tasks. Responses were recorded on a numerical scale with a response of 1 representing strong preference for the *VR* condition, and 7 representing strong preference for the *2D* condition. The questions

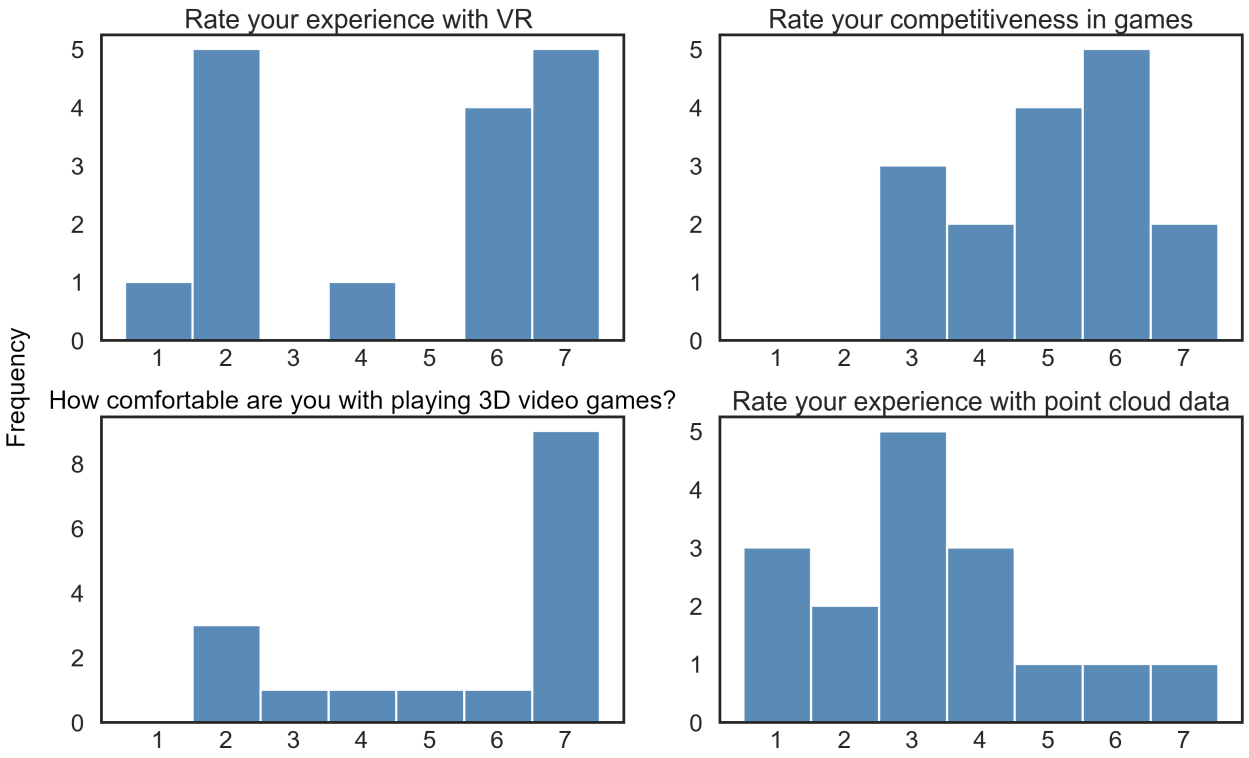

Figure 3.7: Background questionnaire results.

and distributions are shown in Figure 4.9. Using a one-sample two-tailed t-test centered at neutral, all of the results showed significant preference for the *VR* condition at the p<.001 level, except for the question asking about relative speed/efficiency for the counting task, which was significant with p=.03.

The main objective evaluation metric for both the annotation and counting tasks was completion time. In the annotation task, participants were asked to continue the annotation until they reached 98% completion, but intermediate times to 80% and 90% were evaluated as well. These results are shown in Figure 3.9. Generally, there was a learning effect across subsequent trials for within subjects comparisons, but a paired two-tailed t-test between

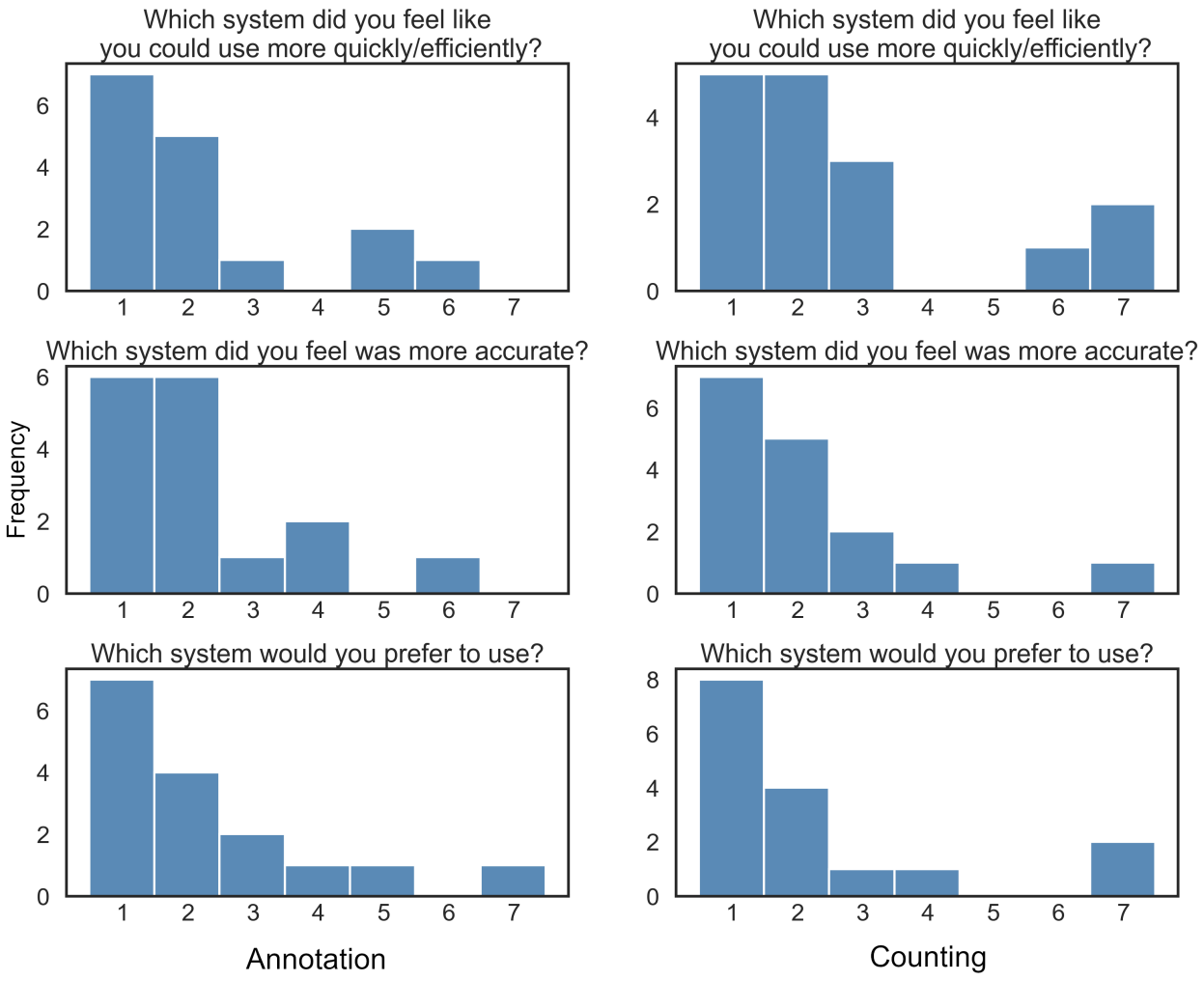

Figure 3.8: Relative preference between the *VR* and *2D* conditions. A score of 1 represents preference for the *VR* system, and a score of 7 represents preference for the *2D* system.

all the first- and second-attempt completion times found no significant difference. In the annotation task using a paired two-tailed t-test, the *VR* condition was found to be significantly faster to 80% (p=.002) and 90% (p=.001) completion, but no significant differences were found at the 98% completion threshold. For 80% completion, times were 55% faster in the *VR* condition (M=43.5, SD=31.7) than in the *2D* condition (M=97.7, SD=69.8), and for

90% completion, they were 51% faster in the *VR* condition (M=62.3, SD=46.1) than the *2D* condition (M=126.8, SD=79.4). No significant differences between the *VR* and *2D* conditions were found in completion times for the training session for either of the tasks using paired two-tailed t-tests.

For the counting task, times were found to be 16% faster on average in the *VR* condition, which was found to be significant (p=.022) using a paired two-tailed t-test. While no true count was calculated to compare accuracy, as such a count would be inherently biased toward one of the conditions depending on how it was calculated, the average counts between the *VR* and *2D* conditions showed a trend towards higher counts in the *VR* condition with p=.051. Participants also performed significantly fewer commit operations in the *VR* condition (p=.008), meaning they kept higher numbers in working memory before entering their subtotals and restarting the count. The results for these measures in the counting task are shown in Figure 3.10.

3.6.1 User Feedback

Free response user feedback was requested after each of the four tasks primarily to discover any usability issues that would not otherwise be captured. In the annotation task, the most common response was the difficulty in visually finding the last few percent needed to achieve 98% completion, with 9 out of 16 users having a similar comment during their first attempt at the task. Two users reported difficulty as a result of depth estimation, and three users found it difficulty to annotate due to occlusion problems with the rest of the point cloud. These reports only occurred in the *2D* condition. No other significant usability issues were

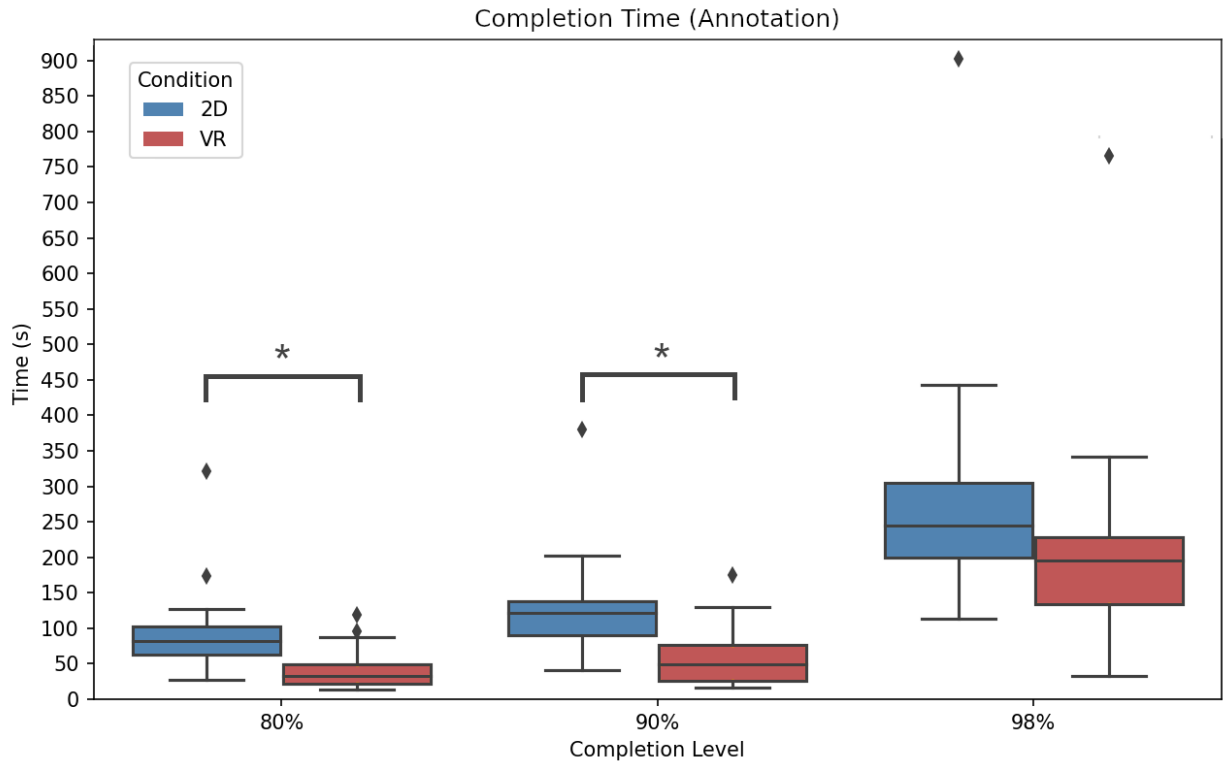

Figure 3.9: Completion times for the annotation task at the 3 target completion levels. Significant differences occurred between the *VR* and *2D* conditions at the 80% and 90% levels using a paired t-test.

reported by participants. After all parts of the study were completed, general feedback was requested. One participant noted that "Precision with keyboard and mouse was a little better, but visual perception was much easier in VR." Another stated, "For the first task I preferred the 2D much more than the second task, where I preferred the VR. I believe that has a lot do with the movement and accuracy needed. ... For the second task the 2d was harder because there was a lot more to pay attention to."

3.7 Discussion

The results largely supported our hypotheses, though interesting nuances arose that were not expected.

H1: Participants will be faster overall with the *VR* system in the counting task. Confirmed. Completion time for the counting task was significantly faster in VR, though we were surprised that the magnitude of the difference was not higher. It should be noted (see H3) that participants also tended to find more cotton bolls in *VR*, likely increasing the amount of time required slightly due to the extra work involved.

H2: Boll counts will be higher and more consistent when counting with the *VR* system. Not confirmed. A trend towards higher boll counts was found in the *VR* condition, but the difference was not significant. Unexpectedly, the variance of the counts between the two conditions was also similar (in fact, count variance was higher in VR). This is an encouraging result, since the purpose of a system like this is for ground-truth gathering, and significant differences between the two conditions would cast doubt onto the reliability of either of the systems. Absolute accuracy of either of the conditions can not be calculated, as there were ambiguities in what parts of the plant should count as a boll. Participants were told to use their own judgement when determining whether to count a half-opened bud as a boll or not, but were asked to be consistent between the *VR* and *2D* conditions with these decisions.

H3: Participants will report higher accuracy and perceived efficiency using the *VR* system Confirmed. Both 7-point scale questions asking about perceived accuracy

as well as speed/efficiency resulted in significant preference for the *VR* condition, even for the annotation task where the difference was not significant. Preference questions were presented as a reflection of the task as a whole, and participants were not aware of the 80% and 90% thresholds for timing used in the evaluation. The reason for this discrepancy in the perceived and actual speed in the annotation task may suggest an altered sense of time caused by a more engaging and novel experience in the *VR* system, rather than the more familiar desktop interface. This effect may also have had an influence on the results of the final hypothesis.

H4: Participants will prefer to use the *VR* system. Confirmed. Both the annotation and counting tasks showed significant preference for the *VR* system. This was further verified by participant feedback.

Other findings: Though we did not form any hypotheses about cognitive load of the task, the commit interface of the counting task offered a way to delve into this aspect. We found that a significantly lower number of commit actions were performed in the *VR* condition than the *2D* condition during the counting task. While not entirely unexpected based on our pilot usage of the system, this showed a willingness of the participants to keep larger counts in working memory compared to the *2D* condition. There are several effects that could have contributed to this result. It is possible that the *VR* condition allowed for a more intuitive and instinctive painting operation, reducing the mental workload of the painting and leaving more room for the actual counting process. Another possible explanation for this effect is the reduction of clear continuity breaks in the *VR* system. In the *2D* condition, every time the camera perspective is changed, the user is presented with a visually different scene with few corresponding elements to the previous image, but in the *VR* condition, perspective is changed

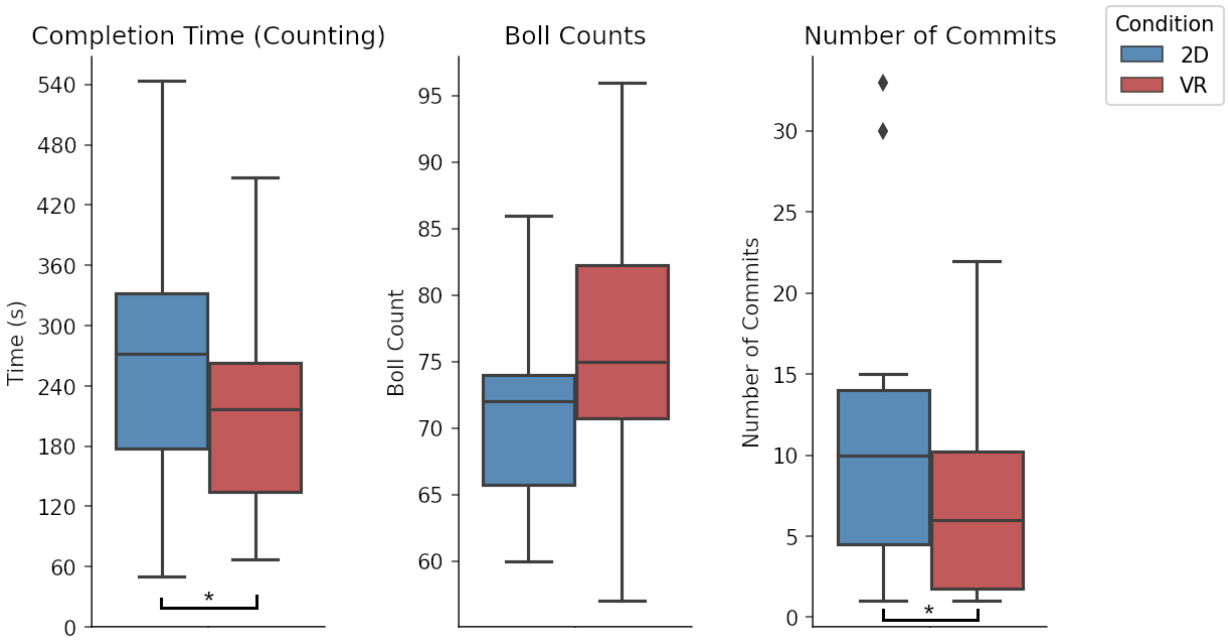

Figure 3.10: Output measures from the counting task. Significant differences were found between the *VR*/*2D* conditions for completion time and number of commits using a paired t-test.

more gradually through head movement and locomotion by hand movement. Combined with the effect the stereoscopic rendering has on perception of the scene as a 3D object rather than a projected 2D image, the *VR* condition is a more continuous experience, while the *2D* system can conceptually be broken into smaller segments of panning movement separated by orbit interactions. These breaks may have provided a better opportunity for entering the subtotals. The last potential reason for the behavior could be increased friction in entering the total using the virtual keypad in VR, though none of the participants reported this as a source of difficulty.

We also investigated the completion times for the annotation task. In an attempt to characterize the completion time curve between the two conditions, completion times were compared at a few checkpoints, including 98%, 90%, and 80%. 98% represents the maximum easily achievable completion percentage as seen in informal pilot trials during development, and 80 and 90% were chosen as regular intervals below this point. Comparing checkpoints much lower than this completion percentage was not expected to provide useful data in the completion of the task, since a single rough painting stroke could already achieve 50-70% F1 score in tests. The majority of participants did not use the same method as predicted by the authors in the *2D* condition. The originally conceptualized method involved painting roughly down the length of the point cloud to cover the target bamboo stem, then rotating the camera by 90 degrees and erasing any areas that were previously not visible and were false positives. However, this method assumed that participants would disable the depth limit feature, which they generally did not do. With depth limiting turned on, the completion trajectory was more linear. In order to get to 90% completion, only a general concept of the structure of the bamboo stem was needed in VR to paint broadly through the center of the point cloud. However, in order to achieve the full 98% dictated by the completion guideline, the task switched to more of a searching problem. The greater angular resolution of the 32" 1440p monitor may have contributed to better performance in the *2D* condition during this part of the task. In addition, it is possible that visual search for small details was inhibited by stereo convergence, which requires not only searching a 2D visual space, but to converge to multiple depth planes.

3.8 Conclusions, Limitations, and Future Work

This work designed and evaluated the potential of an immersive VR system for point cloud visualization and annotation of agricultural LIDAR datasets. The system enabled users to annotate, at the individual point level, large clouds at interactive frame rates in immersive VR with the help of a continuous LOD system that emphasized details at controller locations. We also designed a desktop interface for non-immersed use and comparison in a user study. Our study findings suggest that the immersive VR interface would increase both user performance and user satisfaction, though its efficacy likely diminished during latter stages of completion, requiring more detailed search and manipulation.

Our study had several limitations. First, participants had a relatively high overall level of expertise with VR and 3D modeling programs, game playing and some with experience using point cloud tools. This recruitment was intentionally designed to avoid issues that we may have faced with a lack of familiarity with such tools within the general population, which may have created a bias in our outcomes. This is, perhaps, why we did not find any evidence of nausea from using the VR locomotion interface, though in practice, more comfortable techniques such as real-walking could be used. The gender disparity that arose from this recruitment also limits the generalizability of this work to the general population. Another limitation was that we did not give users multiple attempts with each Task-Interface combination, aside from the training-testing phases of each task. This compromise was made due to the already long study duration and worry about fatigue. It is unclear which interface is fastest to learn, and if additional training would reduce the difference in performance.

Finally, the study compared two hardware interfaces that differed in many ways (stereoscopy, resolution, field of view, controller DoF, etc.), and hence we cannot ascribe the differences in outcomes to particular features. Instead, we optimized the application interface to the available input and output devices, in an attempt to minimize the difference in user performance between the two systems. A lingering issue with this approach is that there is no effective limit to this optimization process. In all likelihood, both interfaces could continue to be improved for the task, though VR interfaces may have more room to grow. For example, we could reduce hand jitter during painting operations in VR, by increasing the control-display ratio, or by adding interactions that allow the user to scale the point cloud dynamically to focus on a specific region at a more comfortable size.

In fact, we see the greatest future value in combining the two interfaces. The current application is able to switch between modes in the middle of a task, and could be made to function simultaneously (two users) to accelerate the work. Already, mixed reality displays are emerging that allow for effective use of planar displays and 3D content. The benefits of both systems could be used to complement each other, and deeper understanding of their corresponding strengths could result in higher quality user experiences in the future.

Chapter 4

Versatile Mixed-method Locomotion under Free-hand and Controller-based Virtual Reality Interfaces[1]

[1]Anton Franzluebbers and Kyle Johnsen. October 2023. VRST '23. Reprinted here with permission of the publisher.

Figure 4.1: Three tasks completed under free-hand and controller-based interfaces within the conVRged social VR application.

4.1 Abstract

Locomotion systems that allow the user to interact with large virtual spaces require precise input, competing with the same inputs available for performing a task in the virtual world. Despite extensive research on hand tracking input modalities, there is a lack of a widely adopted mechanism that offers general-purpose, high-precision locomotion across various applications. This research aims to address this gap by proposing a design that combines teleportation with a grab-pull locomotion scheme to bridge the divide between long-distance and high-precision locomotion in both a tracked-controller and free-hand environment. The implementation details for both tracked controller and tracked hand environments are presented and evaluated through a user study. The study findings indicate that each locomotion mechanism holds value for different tasks, with grab-pull providing more benefit in scenarios where smaller, more precise positioning is required. As found in prior research, controller tracking

was found to be faster than hand tracking, but all participants were able to successfully use the locomotion system with both interfaces.

4.2 Introduction

Visual hand tracking is becoming a standard feature of Augmented and Virtual Reality (AR/VR) systems, with some systems offering an option for hand tracking or controller tracking (e.g., Meta Quest 1/2/Pro, Pico Neo 4) and others relying on hand tracking exclusively (e.g., Microsoft Hololens, upcoming Apple Vision Pro). Eliminating the requirement of controllers is attractive as hands offer a more convenient, always available, and perhaps more natural interface. However, since hands have neither buttons nor tracking aids such as LEDs for more precise positioning, there are considerable challenges in user interface design. This paper discusses how we designed such a free-hand interface, attempting to maintain the versatility of a controller-based interface to support many complex tasks and to keep usability high enough to warrant its use.

A key aspect of this work that differentiates it from other studies on free-hand interfaces is that it is a study built for, *and within*, a real-world social VR application. That application is currently used for a number of other projects, and has an established and versatile controller-based interface already. The free-hand interface was intended as a convenient option for when controllers are not available (e.g., dead batteries) or not desirable, with the understanding that it would likely be less usable due to intrinsic issues in hand tracking (e.g., lack of buttons) and technical issues in tracking the hands with head-mounted cameras. Thus, the primary

purpose of the current work was not to build an equivalently usable system, but instead to minimize and then quantify the usability differences.

We especially focused on locomotion in the design of the interface. The controller-based interface provides teleporting for large distance movements, fixed increment turning, and a grab-pull metaphor for precise positioning, relying on multiple controls for activation and excellent controller position and orientation tracking for control. Providing the same movement versatility in the hand-based locomotion interface required extensive adaptation. Instead of multiple buttons, a single index-thumb pinch gesture was chosen because it was one of the more reliably detected hand gestures and is commonly used in the Meta Quest home interface, increasing familiarity. Together with the pinch gesture, region-based, contextual activation with visual feedback resolves ambiguities and minimizes errors, and filtering is employed to increase accuracy.

After finalizing the design of the interface, we conducted a user study with 20 participants where we gathered detailed performance data while they completed a variety of tasks with the controller-based and free-hand interfaces. In addition to raw performance statistics such as travel time, we were interested in seeing how participants leveraged the various techniques when they had the option. Results from the study, including user feedback, confirmed our expectations that controllers were significantly faster for performing nearly all tasks. However, all users were able to complete all tasks, without reported complaints about simulator sickness and only two reported instances of fatigue. This suggests that the free-hand is a worthwhile addition, one that is likely to continue to improve as hand-tracking quality increases.

4.3 Related Works

There is a wide breadth of research that evaluates locomotion techniques in VR, but only a subset is focused on how they work with hand tracking, though several compare hand tracking to controller tracking for other tasks [52, 83]. Schäfer et al. evaluated four different teleportation techniques designed for use within a hand-tracked environment [100]. Two of these were two-handed techniques, which used a gesture on the other hand to activate the teleport; and two were one-handed, which used a "dwell" technique to activate the teleport. The evaluation task was a long hallway with "primitive graphics style" with 10 columns at random intervals for teleport targets. The authors found no significant advantage for two-handed versus one-handed gestures, concluding that one-handed interaction was sufficient for a usable teleporting interface. However while teleporting was disabled while the hands were near an interactable object, it is not clear whether this the ability to activate the teleporter ray using the tested gestures would interfere with more complex task interaction or social gestures. Work by Zielasko et al. also compared controller-free locomotion methods to a traditional controller, recommending methods with more freedom of movement such as the ability to move backward, though the controller-free methods did not use tracked hand position as the input method for locomotion [138].

Kim et al. explored a variety of non-controller input methods including hand-tracking, eye-tracking, and electroencephalography (EEG) [53]. A user study found "relative superiority of eye-tracking for location targeting and hand-tracking for teleport triggering." Our own study took inspiration from the teleporting task design, which consisted of a hallway with

teleport targets on alternating sides. The spacing of the targets was such that the angle to the next target included a variety of angles and distances, and the alternating design ensured even counts of left and right turns. Both of these two works used an external Leap Motion module for their hand tracking. Visual tracking of hands has only become popular in recent decades, but other systems have used gloves to more accurately track gestures such as the work from Mapes et al. in 1995 [71].

Outside of teleporting, some works look at the use of hand tracking to control the speed or direction of smooth locomotion techniques. Zhao et al. used unique hand gestures for locomotion, including a "finger distance" gesture where the distance between the index finger an the thumb of a hand would determine the movement speed of the locomotion system, or another where the number of fingers displayed would increment the movement speed [137]. Their third system used the frequency of finger tapping for speed control. These systems were compared to an untracked gamepad in a user study performed outside of immersive VR.

Though it is relatively common in popular VR games (e.g. Nock[2], Lone Echo/Echo VR[3], Gorn[4], Brass Tactics[5], Google Earth VR[6], and to a lesser extent Gorilla Tag[7], etc.) hand-based grab-to-move locomotion has a relatively small body of published research [20]. Arm-swinging could be seen as a related technique, but it generally is a mechanism to influence the velocity of a sliding locomotion technique rather than direct manipulation over the user's position in 3 degrees of freedom [131, 13, 88]. Rantala et al. included a grab-pull

[2]https://nock.game/
[3]http://echo.games/
[4]https://store.steampowered.com/app/578620/GORN/
[5]https://www.oculus.com/experiences/rift/1101975213197949/
[6]https://vr.google.com/earth/
[7]https://www.gorillatagvr.com/

system in their evaluation of several locomotion techniques for a visual observation task, and found that it outperformed a non-continuous teleport interface for their task, which involved counting small dots along a hallway as they moved [96]. Lim et al. also compared three locomotion techniques that included grab-pull, but focused on the ability for this technique to move in three dimensions [65].

In addition to locomotion, grabbing and interacting with objects in the virtual world must be designed for a high quality multi-purpose VR application. Schäfer et al. compared a controller to a hand-tracked interface for picking up and placing objects [99]. They found significantly better performance when using a tracked controller, but did not find any significant differences between their proposed hand tracking grab techniques, either a forefinger pinch or a whole-hand grab. Masurovsky et al. and Hameed et al performed similar hand-controller comparisons, and both found significantly better performance when using tracked controllers compared to controller-free interactions [73, 44].

Other works tried to combine the benefits of both controller and controller-free inputs. Capece et al. presented a system that used a tracked controller in one hand for the use of the precise button inputs, and pinch with hand tracking to interact with the nodes on a 3D graph [14]. All locomotion inputs were performed using directional input on the controller.

4.4 System Design

4.4.1 Environment

The work presented here is built in the context of *conVRged*, a multi-purpose social VR world developed in our lab across several research and outreach projects, which is in the process of being converted to a fully open source project[8]. The application is designed such that it can contain a variety of experiences, from presentation-focused environments with screen-sharing, shared whiteboards, to games such as chess, checkers, remote-controlled cars, and a playground. The system consists of several separate environments for each target application. An elevator metaphor is used to travel between environments, with the buttons in the elevator used to select a target environment. While *conVRged* is used within our research group, its design mimics aspects of several commercial applications, such as VR Chat, Rec Room, or Horizon Workrooms.

While the application supports custom avatars (via Ready Player Me[9]), for the study we opted to use an androgynous avatar model with a pure white texture, as seen in Figure 4.1. This avatar was present whether the user is using hand tracking or traditional controller tracking, but instead of using the hand models provided by Ready Player Me for hand tracking on the Quest 2, the standard Meta hand models were used, which match the size and shape of the user's real hands.

[8]https://github.com/velaboratory/conVRged-OSS
[9]https://readyplayer.me

4.4.2 Interaction

Most of the environment-specific interaction in the application is accomplished by a grabbing metaphor, wherein any movable object can be temporarily attached to the virtual hand and then released or thrown. This is accomplished by moving the avatar hand close to an object, which highlights if it can be picked up, and then activating and deactivating a grab intent. When using the Quest 2 controllers, this action is performed with either hand-trigger. When using hand-tracking, it is performed by bringing the tips of the index and thumb finger close enough together (pinching). Additionally, the avatar hands can be used on virtual touch screens by intersecting the virtual hand finger tip with a control or by raycasting from a distance.

4.4.3 Locomotion

The locomotion system complements physical user movement (1-1 with real world movement) with virtual teleportation and grab-based sliding in any direction. The combination of these supports diverse physical and virtual environments and tasks.

Grab-pull

Some tasks, like chess, are performed on a very small scale. In the real world, chess players are generally seated in front of the board, and do not get up or move around much. However, the initial body placement can be difficult, especially if seated, and depends on the height and comfortable reach of the person. In order to enable a comfortable playing position in VR,

precise micro-adjustments must be possible beyond that of a point and teleport mechanism. To achieve this, a "grab-pull" sliding system was implemented. The user can use a button on the controller or a pinch activation gesture in hand tracking mode to lock their hand to the world, then when they move their hand, their body would move to keep their hand in the same location. Momentum is preserved when releasing, so extra distance can be achieved with less effort, but velocity damping is applied to reduce this effect to a maximum of a few meters. Prior research has found small amounts of translation gain can be applied without a perceivable visual effect, but can be used to cover larger distances more easily [136, 133, 130]. A translation gain of 1.5 was applied to this technique for this work. For both the hand-tracked and controller systems, the input for "grabbing" the air to activate the grab-pull mechanic is the same input as grabbing objects in the world to pick them up. If it is not clear to the user if their action will result in motion, simulator sickness may occur [106].

On the Quest 2, hand tracking is performed by machine vision algorithms that process data from four infrared cameras mounted to the headset that are also used to track the controllers. Significant jitter can be observed at all times, with degraded accuracy as the hands become occluded and cut off in the camera view or as they move quickly and become blurred. As a result, several mitigations were developed to reduce the effect of poor hand tracking quality on the locomotion experience. A double-exponential smoothing filter was applied to the raw pinch position at all times. In addition, extra smoothing was applied inversely proportionately to the distance to the headset once the hands were within a certain radius. A common action when using grab-pull to move forwards results in the user's release point ending close to their face. The accuracy at the release point is the most critical, since

it also determines the distance traveled due to momentum. By increasing the smoothing at this point, release velocity became more consistent for this type of motion.

Teleportation and Snap-Turning

Other tasks require the user to cover larger distances quickly, such as forming small groups for conversations or playing a larger-scale game. Using grab-pull can only translate by a few meters for each action, so a faster method is useful. In addition, users seated in non-rotating chairs must have a way to virtually turn their avatars. Both of these can be supported by a mechanic that transforms the tracking volume to achieve the desired offset or rotation [12].

In the application, teleportation is achieved by first activating the teleporter. When active, a parabolic trajectory is visualized emerging from an origin and ending at a target position somewhere in the world. If the target position is valid, deactivating the teleporter instantly transports the user to that location. Snap turning is a single activation, left or right, and results in a corresponding 30 degrees rotation of the tracking volume about one of two positions. When the grab-pull system is active, the center of rotation is the hand that is grabbing; otherwise, the play space rotates around the head position.

When using controllers, teleportation is activated by the thumbstick, as is commonly done in consumer games. Holding forward on the thumbstick activates the teleporter arc, which emerges from the controller position. Releasing the thumbstick activates the teleport. Snap rotation activates when pushing the thumbstick left or right, which must be repeated for multiple rotations.

For hand-based teleportation, the same arc-based teleporting ray was used, with the only difference lying in the activation and pointing of the ray. To activate a teleport, the pinch action was reused, as it was found to be the most reliable gesture. However, as it was already in use for grab-based sliding, we needed to multiplex the two actions. Our solution was to reserve three portions of the reachable space for activating the teleporter and snap turns. These regions could not be placed near the center of the viewport since they would interfere with other tasks in the virtual environment, such as picking up objects. The bottom of the viewport is also an inconvenient area to put regions because the hands are often near the bottom of the screen when resting. Therefore, the teleport region was placed forward and above the user's head position, and the snap-turn targets were placed to the sides. Pinching while the user's hands are inside a sphere of a 10 cm radius activates the teleporter, and the user can then lower their hands to point the ray at the desired location on the floor. The direction of the ray is determined by the vector between a point near their head and their hand. If the exact head position were used in this calculation, the ray would be difficult to see since the user is looking directly down the length of the ray. Instead of the head position, a point to the side (depending on which hand activates the teleporter) and below their head was used in this calculation to reduce fatigue by moving the typical release point to a more natural hand position. The choice to use the head-hand vector for the teleporter ray direction differs from the implementation for controllers, which uses the controller direction. This difference was necessary because the accuracy of the orientation of the controllers is far higher and more temporally stable than that of tracked hands due to the presence of the inertial measurement unit (IMU) in the controllers, while the hands rely on purely visual tracking. Additionally,

the goal of this study was not to compare hand tracking precision with controller tracking but rather to investigate a realistic implementation of a similar locomotion system for both interaction mechanisms. We believe the chosen inputs represent a high-quality locomotion mechanic that does not interfere with the ability to interact with the world for complex tasks.

4.5 Study Design

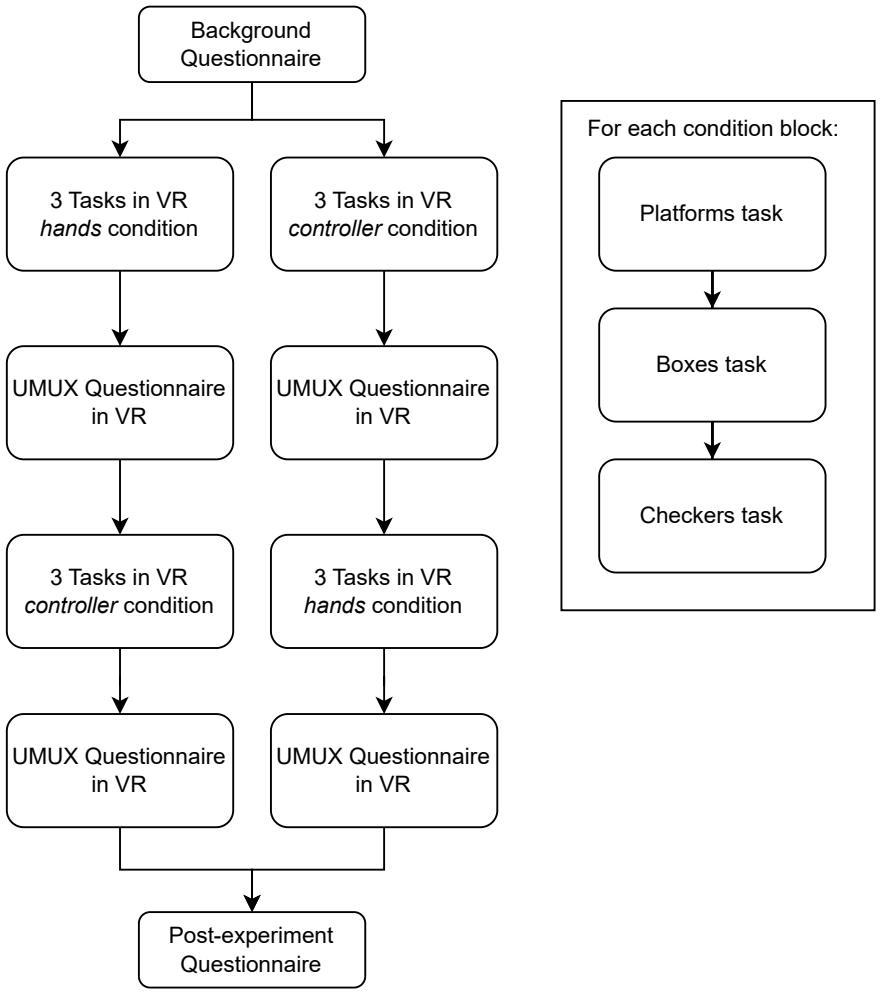

Figure 4.2: Study procedure. The two participant groups varied in order of the *hands* and *controllers* conditions

We conducted a user study to evaluate the proposed locomotion technique across both hand-tracked (*hands* condition) and controller-tracked (*controllers*) conditions as well as through a variety of tasks. A within-subjects, crossover design was used, such that each participant used both the *hands* and *controllers* condition for each of three tasks as seen in Figure 4.2. The input condition was alternated between participants, but the three tasks were always performed in the same order first for one condition, then for the other. This was done to combat the learning effect that is common in VR user studies. Performance comparisons between the tasks were not intended, so no order variation was needed. At the beginning of each input condition, participants were instructed how to use each of the locomotion inputs, as well as grab and place a single object. This tutorial procedure took approximately 30 seconds, after which they proceeded to the first task. In general the goal of developing the study environment was not to build a bespoke apparatus for this study, but rather to build a world with various tasks that the user was guided through by the experimenter, who was also in VR with the participant in the same virtual space. For example, there were no artificial limits on where the participant could teleport in the platforms task, and the grab-pull locomotion system was not disabled. Participants were just told that their task was to teleport to each platform in turn, just as would happen in the real world.

4.5.1 Tasks

Three tasks were designed to test the system in a variety of situations. Each of the tasks represents a different segment of possible application usage.

Platforms

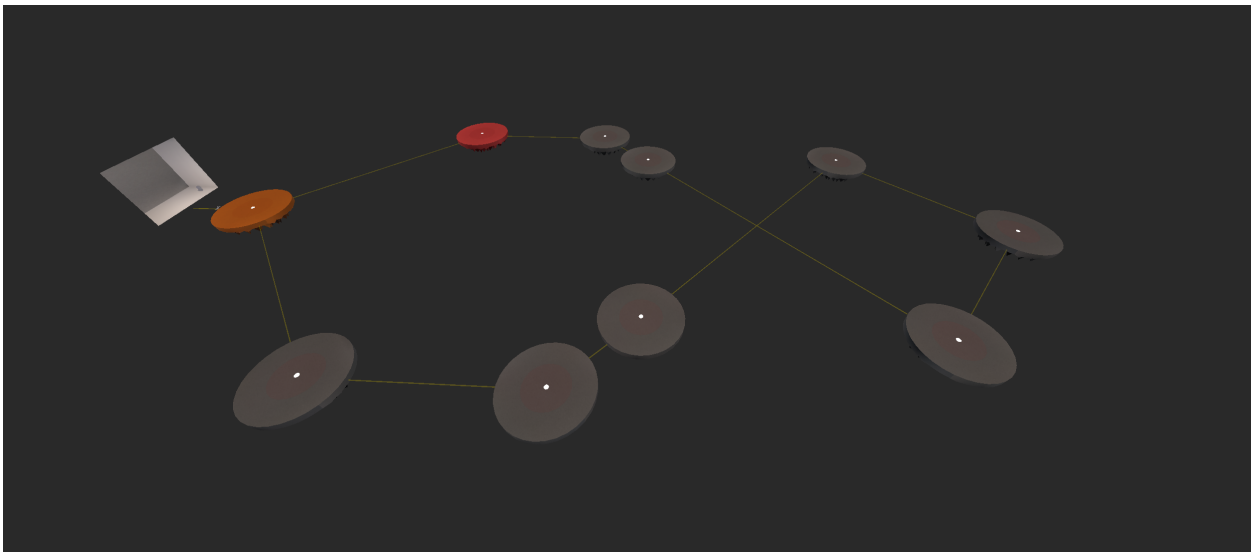

Figure 4.3: An overview of the platforms task. The layout contains varying distances and turn angles.

The first task was intended to focus on measuring the precision and speed of the teleporting portion of the locomotion system. The task consisted of 10 platforms of 2 m radius (Figure 4.4). All of the platforms were at the same and arranged in such a way that there were a variety of left and right turns of various degrees as well as a variety of distances, from 10 m to 36 m. See Figure 4.3 for the layout. Due to the parabolic nature of the teleporting ray in both the *hands* and *controllers* conditions, the 36 m distance was near the maximum comfortably possible for the interface. Participants were instructed to perform the course twice in a row. The first time as "precisely as possible" and the second time "as fast as possible," in order to encompass both ends of the speed-precision trade-off. The "fast as possible" trial was always performed second to match the natural increase in speed that happens with growing familiarity with a task. While the grab-pull locomotion technique was

not forbidden verbally or through the software, the task was described as teleporting to each platform, and only teleporting data was analyzed.

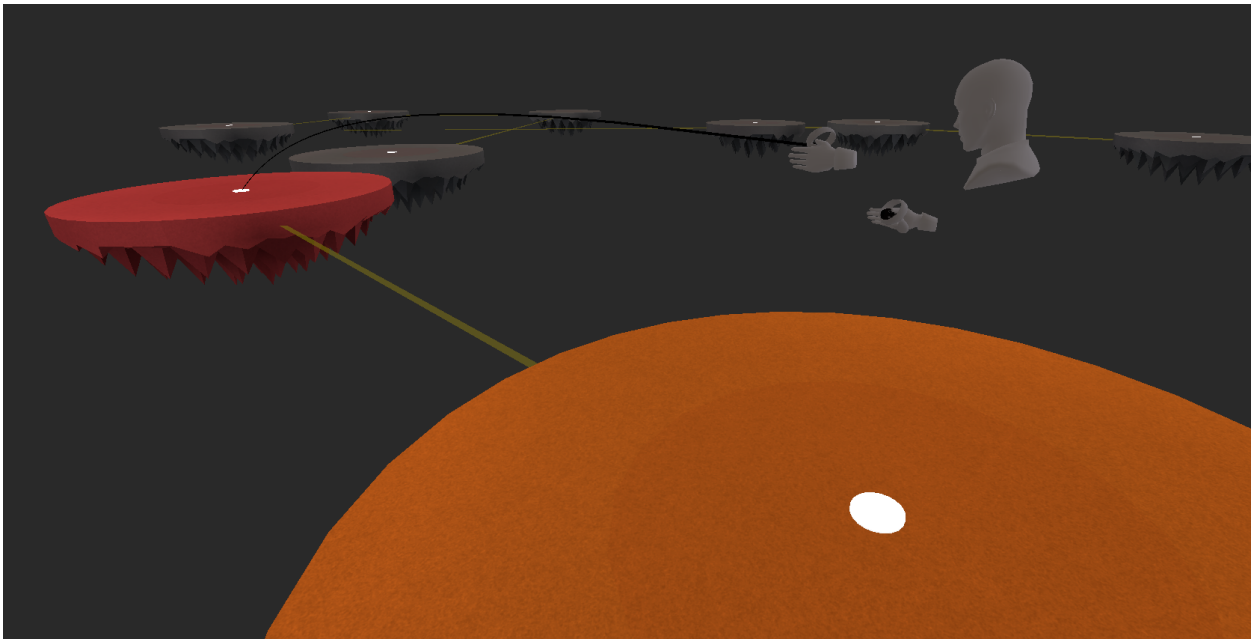

Figure 4.4: A user teleporting from one platform to the next. The current platform was highlighted in orange, and the next platform was highlighted in red.

Boxes

After completing the platforms task, participants looked to the center of the room, where a set of 10 boxes on posts were spawned (see Figure 4.5). This task was designed to mimic similar tasks from related works[84, 29] that also attempted to analyze the task performance of various locomotion techniques. One addition beyond that of related work is to randomly set the elevation of the boxes to any value within 1 to 2 m above the ground. There were two predefined random layouts of boxes, in which the orientation, position, and elevation of the boxes in the room were decided by fixed seeds. The boxes were at least 1 meter apart from

Figure 4.5: A user opening a door of one of the boxes in the boxes task. This is the second of the two box configurations.

each other and were spread in a space of 14x14 meters. To complete the task, participants had to open the door found on one side of the box by grabbing and holding it open while they removed the sphere with the other hand. It was not possible to see whether a sphere had already been removed from the outside, so part of the task was to maintain spatial awareness throughout usage of the locomotion system to travel between boxes. Each box was 60 cm in width, and the spheres inside had a 10 cm radius.

Checkers

The third task was to clean up a messy checkers board as seen in Figure 4.6. Participants were presented with a small side room with a checkers board on which the black pieces opposite of them perfectly aligned, but the red pieces were scattered around the board and floor. The

Figure 4.6: The checkers board in its "messy" state. Several pieces started on the floor.

task was to pick up all 12 red pieces and place them onto the appropriate squares of the board as demonstrated by the black pieces. There was a physical timer on the side of the board with a start and stop button that the participant started and stopped by themselves. The timer provided motivation to complete the task faster as well as providing another precise UI interaction as part of the task.

4.5.2 Study Population and Environment

Following human-subjects review approval, 20 participants were recruited from the local university population and lab groups to participate in the study by direct invitation and word-of-mouth. A $15 gift card was used as an incentive, and the entire experience could be completed in 30-45 minutes. Most participants completed the study in our laboratory, but one participant performed the experiment remotely. Communication with this participant happened through a voice call for the portions outside of VR, and used the built-in voice communication in the VR application for communication during the study. No other differences in the run of the study were experienced. All participants were seated in a fixed-rotation chair as in the work by Huber et al. and used a Meta Quest 2 headset [47]. The *controllers* condition was completed with the standard Quest 2 controllers and the *hands* condition with the built-in hand tracking on the Quest 2 using the "MAX" frequency option for hand tracking V2.0 for the best experience on this platform. The experimenter was also in VR to walk the participant through the study, though they always used the controllers. Frame rate inside the application was consistently 90 Hz with no noticeable frame drops.

4.5.3 Procedure and Measures

After signing the informed consent document, participants filled out a background survey with questions about demographics and prior experience with VR. The necessary buttons or gestures were shown outside of VR for their first condition, then they were instructed to put on the headset and adjust it for a comfortable viewing experience. Once both the participant

and the experimenter were inside VR in the same virtual environment, the experimenter demonstrated and explained both grab-pull, teleporting, and snap-turn locomotion techniques. In the *hands* condition, the participant was first asked to calibrate their arm length using the option in the floating tablet interface. After selecting the calibration option, a small box appeared in front of them, which they were told to grab and release at their maximum arm extension. The participant was observed while completing each of the locomotion techniques at least once, and if they were able to successfully complete them, they were instructed to grab a small cube from a table and put it into a box before selecting the separate scene where the actual study would take place. The experimenter guided the participants through each of the tasks, observing their progress just as if the tasks were completed in a physical lab instead of a virtual environment. Every action performed in VR, such as teleport events, grab events, etc., was logged to a file and used for and used to generate the quantitative results for each of the tasks. After completing the third task in VR, a web browser embedded in the environment was used to present survey questions without the context switch that leaving VR would necessitate. These questions followed a standard Usability Metric for User Experience (UMUX) questionnaire [9]. Between the *hands* and *controllers* conditions, the participant was instructed to take off the headset for a few minutes to reduce fatigue from wearing the headset for extended periods of time as well as to provide the same brief introduction to the buttons or gestures for the second condition. After both conditions were completed, another survey was completed at a desktop computer, asking about comparisons between both conditions and their general experience.

4.6 Results

All participants were able to complete all of the tasks successfully, and no motion sickness, vision problems, or discomfort were reported during the study. The average age of the 20 participants was 22 years (SD=4.2), and 14 reported male, 5 female, and 1 other/prefer not to say. Experience levels were mixed, with 6 participants never having used VR, and 10/20 users reporting 4 or above on a 1-7 scale of VR experience. Comfort playing 3D video games and self-reported competitiveness were similarly mixed.

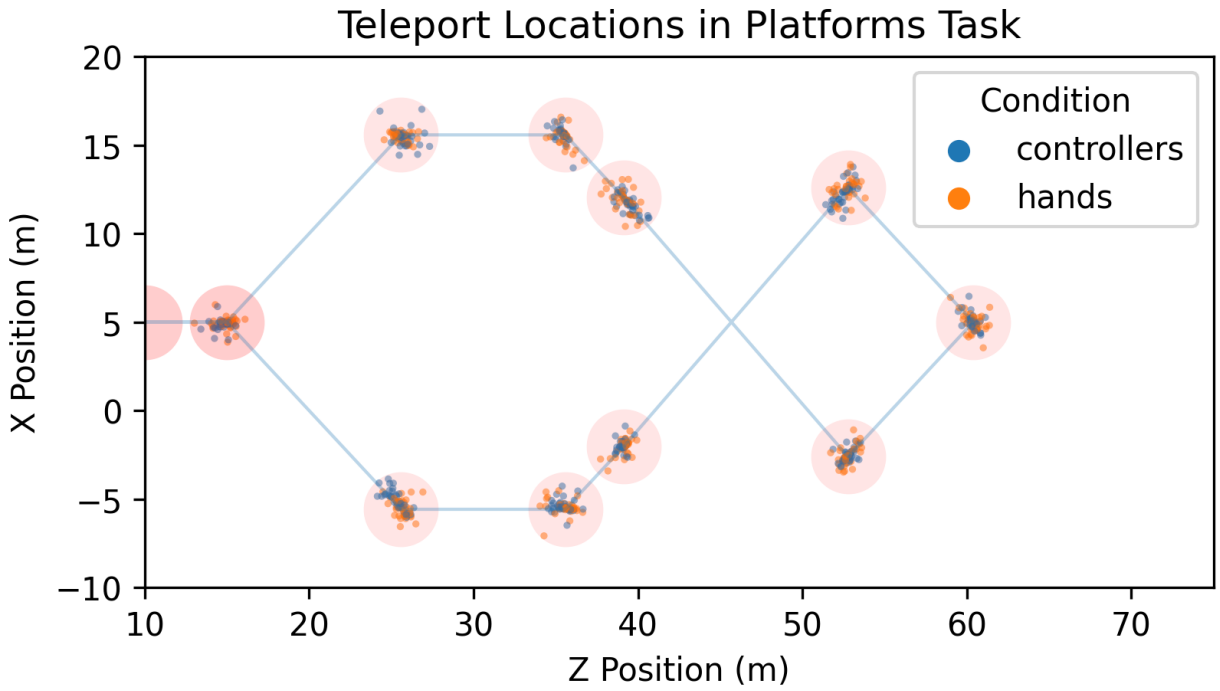

Figure 4.7: Top-down view of the teleport locations in the *platforms* task. The highest error was in the direction of travel.

Using a Chi-square test for independence, no significant differences were found between the two groups assigned to different condition orders. In addition, none of these background

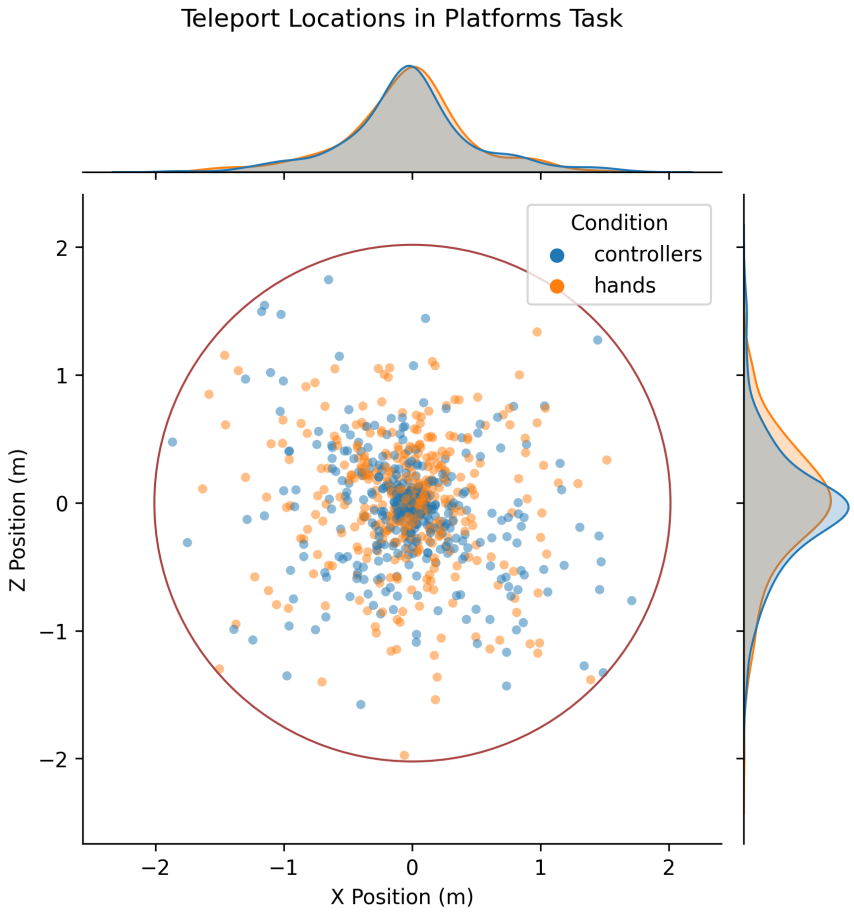

Figure 4.8: Top-down view of all of the teleport locations onto the platforms. Some platforms were approached from different directions, so the direction of highest error is mixed.

results correlated strongly with the performance results, so they were not used as covariables in subsequent analyses.

A Usability Metric for User Experience (UMUX) questionnaire was asked after both the *hands* and *controllers* conditions. UMUX is a four-question survey designed to generate results comparable to the popular System Usability Survey (SUS) while requiring fewer questions to complete [8, 9]. The *hands* condition generated a SUS-comparable score of 0.62, and the *controllers* condition generated a score of 0.78, which were significantly different

according to an independent samples two-tailed t-test ($t(18)=-2.39$, $p=0.02$). No difference was found when all of the first trials were compared to the second trials ($M1=0.71$, $M2=0.68$, $t(18)=0.59$, $p=0.56$).

After both conditions were completed, participants answered four 7-point scale comparison questions between the two conditions as well as two questions between the two primary locomotion inputs. Results for these questions can be found in Figures 4.9 and 4.10. Using a one-sample t-test for variation from the mean centered at 4, each of the answers was found to be significant in favor of the controller condition. A Shapiro-Wilk test for normality found significant deviation from normality ($p<0.05$) in all questions except for the third general condition preference questions (as seen in Figure 4.9 and Table 4.1), and the locomotion system preference question for the boxes task. Each of these responses with non-normal distributions were also tested with the Wilcoxon signed-rank test centered at a value of 4, and were found not to be symmetric about zero ($p<0.05$).

Two comparison questions asked about relative preference between the grab-pull and teleport inputs (Figure 4.10). For the boxes task, no significant preference was found, as participants used a mix of both inputs, but for the checkers task, significant preference was found for the grab-pull input ($t(19)=10.41$, $p<0.001$).

Quantitative performance was also measured in each of the three tasks. In the platforms task, accuracy was measured as the distance from the center of each platform for each teleport. Overall, No significant difference between the *hands* ($M=0.56$ m, $SD=0.42$) and *controllers* ($M=0.53$ m, $SD=0.45$) conditions was found ($t(18)=0.97$, $p=0.33$), and no improvement was found across the first ($M=0.54$ m, $SD=0.43$) and second ($M=0.56$ m, $SD=0.44$) trials

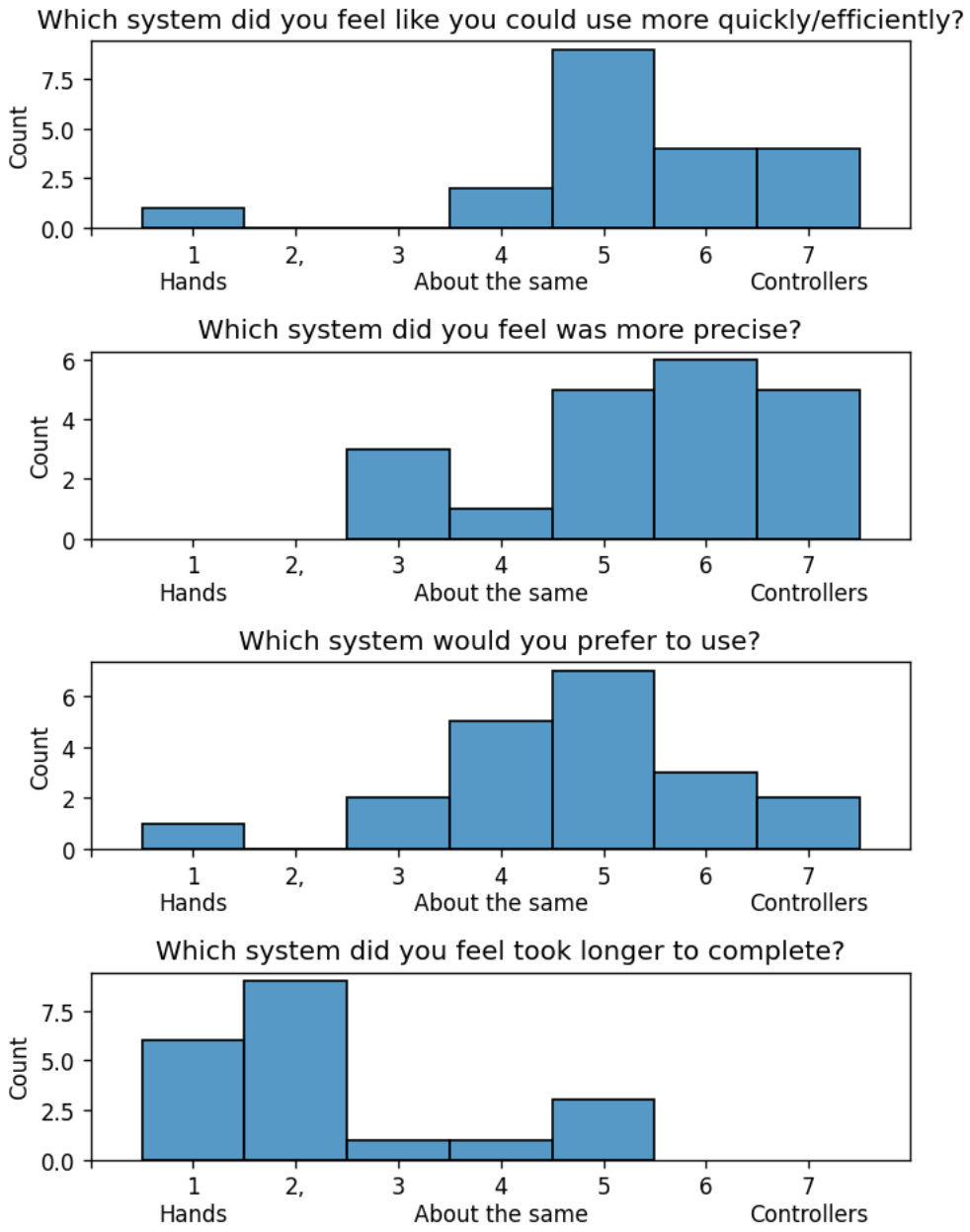

Figure 4.9: Relative preference scores between the *hands* and *controllers* conditions from the post-experiment questionnaire

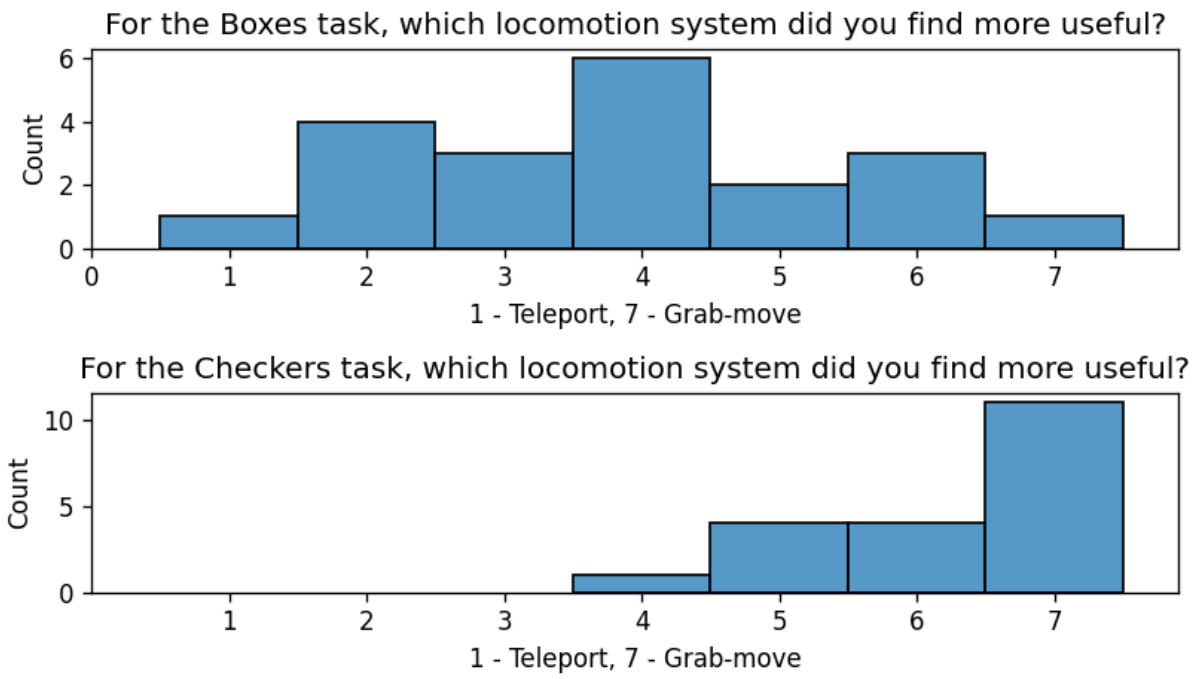

Figure 4.10: Relative preference scores between Teleport and Grab-move from the post-experiment questionnaire

(t(18)=-0.60, p=0.55). When isolating the data for only the first run where the participant was instructed to be as precise as possible, significant difference was still not found, but the p-value was reduced to 0.058 (t(18)=1.90) in favor of the *controllers* condition using an independent samples two-tailed t-test. The data for this accuracy measurement can be seen in Figures 4.7 and 4.8.

Overall completion time was measured by calculating the time between each of the teleports in the sequence. The *controllers* condition (M=2.93 s, SD=2.82 s) was found to be significantly faster than the *hands* condition (M=6.06 s, SD=5.24 s) using an independent samples two-tailed t-test (t(18)=9.32, p<0.001), and this trend remained when isolating the

Table 4.1: Summary statistics for all 7-point Likert scale preference questions asked after each condition or after the experiment. 4 was "about the same" and the ends were in favor of either condition. These results can also be seen in Figures 4.9 and 4.10

Question	1-samp. t-test
For the Boxes task, which locomotion system did you find more useful?	M=3.85, SD=1.63, t(19)=-0.41, p=0.69
For the Checkers task, which locomotion system did you find more useful?	M=6.25, SD=0.97, t(19)=10.41, p<0.001
Which system did you feel like you could use more quickly/efficiently?	M=5.30, SD=1.38, t(19)=4.21, p<0.001
Which system did you feel was more precise?	M=5.45, SD=1.36, t(19)=4.78, p<0.001
Which system would you prefer to use?	M=4.70, SD=1.42, t(19)=2.21, p=0.04
Which system did you feel took longer to complete?	M=2.30, SD=1.38, t(19)=-5.51, p<0.001

data to the "fast" run only. No differences (t(18)=-0.81, p=0.42) were found in completion time between the first (M=2.78 s, SD=2.36 s) condition and the second (M=3.00 s, SD=2.64 s) condition, showing no significant learning effect. In order to perform a teleport, a ray must first be activated, then aimed and released. The time with this ray active was measured for both conditions, and again the *controller* condition (M=1.42 s, SD=1.06 s) was found to be significantly faster than the *hands* condition (M=2.84 s, SD=2.27 s, t(18)=10.78, p<0.001) both overall and for the "fast" run only. When limiting "fast" run only, these times reduced to 0.8 s (SD=0.5) and 1.6 s (SD=0.7), but the ratio remained the same. These results can be found in Figure 4.11.

Completion time for the boxes task was calculated similarly to the platforms task, with the average time between each box as the primary metric. The *controllers* condition (M=12.42 s, SD=7.75 s) was found to be significantly faster than the *hands* condition (M=16.53 s,

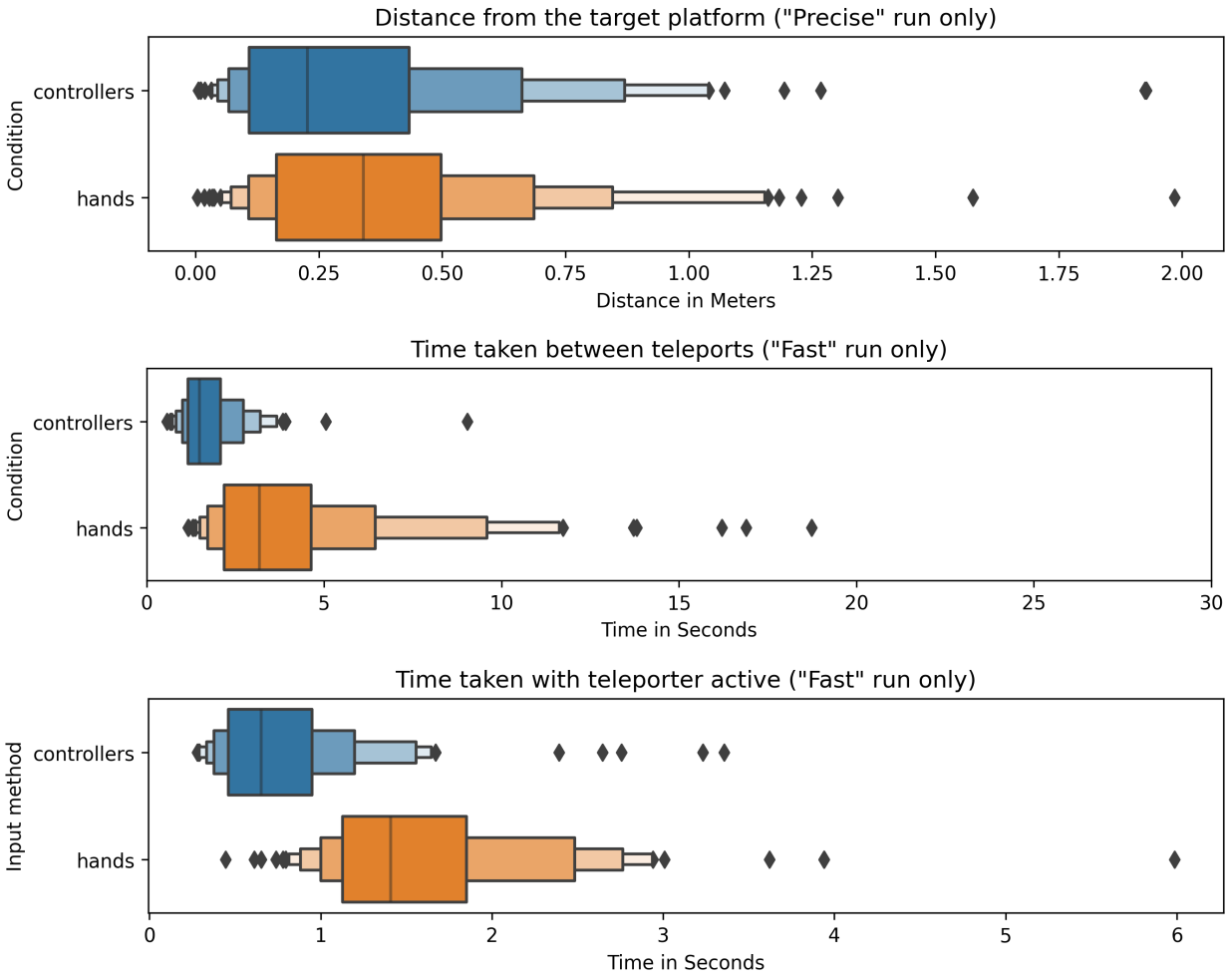

Figure 4.11: From top to bottom: Distance from the center of the platform for each teleport; Total time between each teleport event. This can be multiplied by 10 for the total task completion time; Time with the teleporting interface active.

SD=8.48 s) using an independent samples two-tailed t-test (t(18)=-4.52, p<0.001) (Figure 4.12), and and no difference was found between the first condition and the second condition (t(18)=1.43, p=0.15).

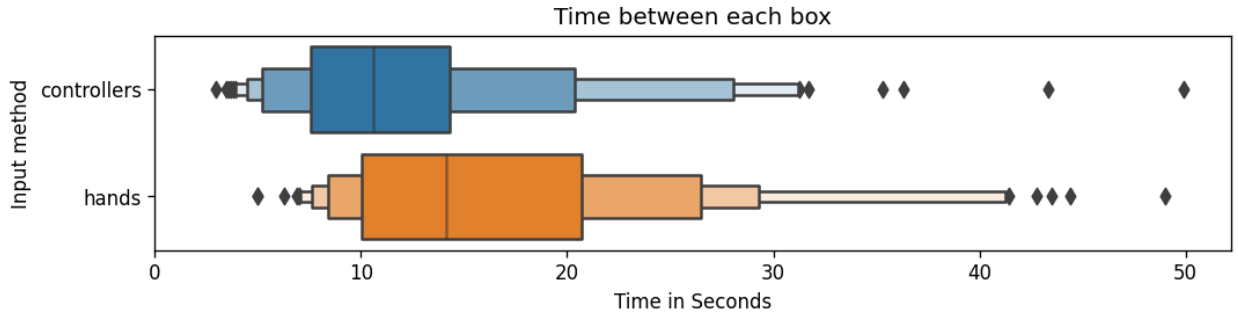

Figure 4.12: Completion time of the boxes task. The time between each box can be multiplied by the number of boxes for total task time.

Completion time for the checkers task was measured by the physical timer operated by the participants. No significant difference between either the *hands* (M=96.37 s, SD=43.18 s) and *controllers* (M=75.92 s, SD=30.89 s) conditions (t(18)=-1.60, p=0.12) (Figure 4.13) or the first (M=86.31 s, SD=33.82 s) and second (M=84.85 s, SD=42.80 s) trials were found (t(18)=0.11, p=0.91). Overall average completion time was 86 seconds (SD=39). As each item was picked up and placed in the correct location, the time spent holding each checkers piece was also recorded. Participants held the pieces 0.6 seconds longer on average in the *hands* condition (t(18)=-3.38, p<0.001), but no difference was found between the first and second conditions overall.

On average, participants completed 36.4 (SD=19.4) grab-pull actions and 9.4 (SD=4.5) teleport actions during the boxes task. These averages changed to 43.8 (SD=17.2) grab-pull actions and 2.6 (SD=2.7) teleports for the checkers task.

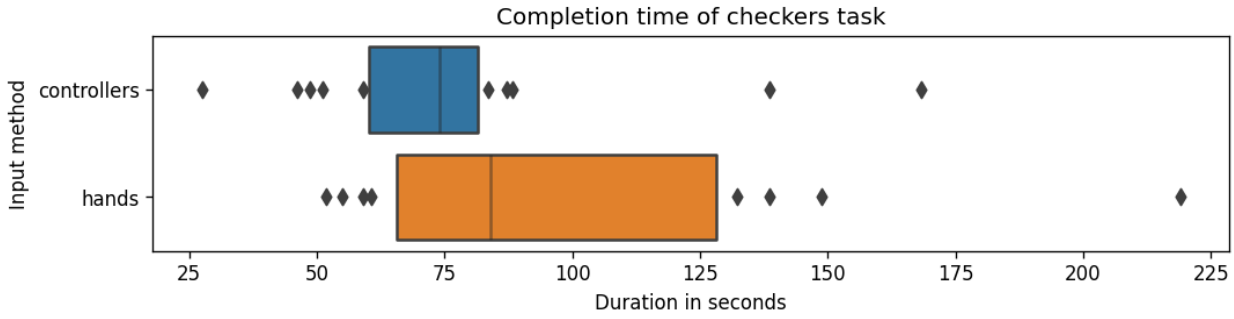

Figure 4.13: The total completion time for the checkers task. This time is the result of the in-application timer operated by the participant.

4.6.1 User Feedback

Free response feedback was requested for several questions in the post-experiment survey to capture issues or comments not otherwise quantified in the quantitative data or other survey questions. The first such question asked participants to explain why they preferred the teleporting or grab-pull inputs for both the boxes and checkers tasks. 15 out of the 20 participants wrote that the teleport was more useful for long distance movements and the grab-pull input was useful for smaller movements or adjustments. One participant commented, "teleporting was very useful in covering a lot of distance very quickly, but I felt disoriented trying to do it quickly and turning after teleporting was always needed. The grab-pull felt better for moving short distances and it felt more natural physically moving in that way." Only three participants commented on the ability for the grab-pull locomotion to move vertically, whereas the teleporting system could only cover horizontal distance. Five people commented that grab-pull was more natural than teleporting, and six mentioned that grab-

pull provided more precise control than teleporting. Another user commented, "Grab and move felt very natural and allowed me to maneuver precisely in the environment." Only one person mentioned fatigue when comparing the two input schemes. None of the responses indicated that the participants perceived the translation gain present in the grab-pull scheme.

Another free-response question prompted participants to discuss the aspect of the tasks or system they found the most difficulty with. The most common issue (seven participants) was the low quality of the hand tracking itself, which resulted in issues performing reliable grab or teleport operations. Five participants found the *hands* condition unreliable at times, especially for the grab-pull technique. Two users mentioned that the high position of the teleport and snap-turn targets in the *hands* condition caused arm fatigue in the platforms task, and two participants commented on the imprecision of the teleporting release, "as I released my fingers to teleport, the teleportation destination shifted whereas with the controllers I did not have this problem."

4.7 Discussion

Overall, the results showed significant preference and completion time advantages for the *controllers* condition over the *hands* condition, which supports prior work in this area, such as that from Schäfer et al., Masurovsky et al., and Hameed et al. [99, 73, 44]. This research focused more on quantifying that performance delta, and we found 20 to 30% faster completion times for the *controllers* condition.

The three tasks were designed to span the range of applicability of the grab-pull and teleporting interfaces, and we saw this affinity reflected in the results. The platforms task was designed to be the best for teleporting, but the task specified the use of teleporting for movement, so the relative usage in that task is not useful. However participants were able to choose which locomotion system they preferred to use in the boxes and checkers tasks. The grab-pull interface had far more invocations than teleports in both tasks, but the teleporting action results in a longer distance traveled. In the box task, a user could choose to teleport from one of the boxes to another, or use several smaller grab-pull actions to cover the same distance. Since the average number of teleport actions was 9.4 for this task, and only 11 out of the 20 participants used 9 or more teleports (the minimum number of teleports required to travel between 10 boxes), the data supports that many of the transitions between boxes were completed purely using the grab-pull technique. This is corroborated with the qualitative observations of the experimenter, who was watching the participant complete the tasks. Part of the task was remembering which of the boxes had already been visited, as it was not possible to see from the outside whether a box had been completed or not. It was relatively uncommon for a participant to forget which boxes had been visited, but several participants did spend some time moving between boxes to find the one they had not yet completed. Since this only happened a small number of times, it was not possible to quantify which of the input methods resulted in higher occurrences of disorientation.

The average number of teleports in the checkers task decreased compared to the boxes task, while the number of grab-pull actions increased. This supports our hypothesis that grab-pull is a more useful locomotion system for manipulation-heavy tasks that do not require

a large range of motion. Preference scores and free-response comments from participants also heavily supported this trend. One behavior that was observed during the study was the use of teleport as a height-reset mechanic. In the checkers task, four of the pieces were on the floor, so participants used the grab-pull technique to move themselves down to the floor to avoid reaching down physically, which would have been especially difficult in a seated position. This advantage in the ability to move more easily in three dimensions is supported in prior work from Lim et al. [65]. When returning to the checkers board to place the piece in the correct location, some participants teleported instead of returning using the same grab-pull technique. The teleport scheme always results in a final vertical position where the physical floor is at the same height as the virtual floor, which is an appropriate height to stand in front of the checkers board to place the piece.

Another trend observed in both the boxes and checkers task was that even in situations where the travel distance was far and the teleporter was used, the grab-pull system was still used to adjust the final position. This would have likely been less common for simpler tasks that required less precision to complete, but the boxes task in this study was specifically designed to require both large-scale movements as well as two-hand interactions at the scale of several centimeters. This was especially noticeable when participants moved to a box, tried to open the door and remove the sphere, but were not able to, then adjusted their position and orientation again using the locomotion system, after which they could more easily complete the task. Qualitative participant responses aligned with this observed behavior.

Another significant difference we found between the *controllers* and *hands* conditions was the increase in usage of the snap-turn mechanic for the *controllers* condition. This difference

can likely be attributed almost entirely to the increase in distance the hands need to move to activate the feature. Using controllers, snap turn is activated by pushing the thumbstick to the side, which is a total movement of only a few centimeters by the thumb. For the *hands* condition, the snap-turn activation target is generally tens of centimeters from the current hand location. The interesting result from this data is how much participants chose to not use snap turn as a result, possibly indicating that most of the snap turns in the *controller* condition were not essential to completing the task, though it is still possible the use of snap turn did contribute to a better user experience. Application developers have a large amount of control over how much this feature is used based on the friction that is required to activate it. Snap turns can be avoided entirely by allowing fully physical rotation, but in this study the participant was seated to avoid this.

4.8 Conclusions, Limitations, and Future Work

In this work we present a mixed-method locomotion scheme that uses teleportation, snap-turn, and grab-pull locomotion to achieve high task performance in a variety of workloads. We conducted a user study in which participants used this locomotion scheme for three different tasks for both hand-tracked and tracked-controller environments. We analyzed the usage patterns across each of the tasks and across both input conditions, and found significant overall preference for the *controllers* condition as well as faster completion times, which aligns with prior work in this space, though accuracy was less affected. More importantly, we quantified the performance delta that switching to a hand-tracked system incurs.

Though we developed a user study that looked at three tasks spanning a variety of usage patterns, not all use cases are covered by this test. It is possible that some scenarios would result in a larger shift in performance deltas for the *hands* and *controllers* conditions that is not anticipated. Future work could expand this evaluation of this locomotion scheme to further tasks. Another limitation of this study is the relatively small size of the study population and the gender disparity. Future VR applications will be used by all members of society, not just the demographics currently more likely to become involved in a VR research study [91].

The overall performance and quality of hand tracking locomotion systems is highly dependent on the quality of the tracking itself. Our system used Hand Tracking V2.0 on the Meta Quest 2, which is only one of the available tracking systems, which may limit the generalizability of our results. Future headsets or software revisions may also have higher quality hand tracking implementations, changing the results of the study by reducing tracking errors. Alternatively, finger and wrist-worn input devices may offer a compromise between hand-tracking and controller-tracking and should be explored as a means to mitigate hand-tracking and pinch activation issues while maintaining convenience.

Chapter 5

Building a research-ready social space

Previous chapters have focused on the development and evaluation of larger systems as part of the larger cohesive whole that is *conVRged*. This chapter instead focuses on the individual reusable components that make up the software used for each of the previous works. For each of these components, the decisions prompting their development, the design choices made during development, and their value to the creation of the larger whole is discussed. These components will perhaps be the longest-standing software contribution of this space of this dissertation, with many of them already being used in projects outside the scope of the work presented in this dissertation for application areas that they were not initially designed for.

5.1 Networking - VelNet

Building a social application of any kind requires a real-time communication system in order to facilitate interaction between users. However, VR increases the demand for bandwidth and latency for several reasons. First, VR applications require a high frame rate to avoid motion sickness and generally have a higher resolution than common desktop or mobile screens. This higher framerate means that inaccuracies in networked data are more visible than in applications running at lower framerates or resolutions. Players in VR applications are also generally represented using more expressive avatars, with elements such as independently moving hands, fingers, or even facial expressions driven by the sensors on the headset.

5.1.1 Security vs. Iteration Speed

In competitive commercial games, a common problem is preventing players from using cheats or hacks to gain an unfair advantage. This is often done by moving as much as possible of the gameplay logic to the server and only allowing the client to send their inputs to the server, which then sends the updated state of the game back to the client. Server authority is a good way to prevent cheating, but it can also introduce latency as the client has to wait for the server to respond before it can update the game state. Latency can be mitigated by predicting the server's response on the client side and resolving inconsistencies when the server's response differs from the expected value. This architecture requires the server to be complex, with knowledge of the interactions and design of the client application. When developing an application with a dedicated server, every time a change is made to

the network behavior of the client, a new version of the server must be deployed, slowing down development by increasing iteration time during testing. Another disadvantage of this architecture for smaller-scale research applications is the cost of the server itself. Dedicated server architecture requires a server with significant processing power, since the server is at minimum simulating the game, and possibly running calculations that verify the integrity of multiple clients' data. One mitigation to the iteration speed and cost tradeoffs is for one of the clients to act as the server, and for the other clients to connect to it. Without a separate central server, this would require the host client to expose their IP address to the other clients, which is a security risk and requires a manual step to port-forward. Relay servers solve this problem by acting as a middleman between the clients, but this increases latency while still requiring the cost of a central server, though this server is generally cheaper than a dedicated simulation server.

5.1.2 Alternatives

During the early development of the application later known as conVRged, we used Photon PUN 2, which uses a client-authoritative model. In this architecture, the central server is not responsible for verifying the integrity of the game state, but instead acts as a pub-sub (publish and subscribe) broker for messages between clients. This means that the server does not need to simulate the game, and can be run on lower cost hardware. However, this also means that the server cannot prevent many kinds of cheating, as the client is responsible for updating the game state. Photon PUN 2 is a popular choice for Unity developers, as it is easy to set up and has a free tier that is often sufficient for small-scale applications[94]. This ease of use

stems primarily from the incredibly simple architecture enabled by the client-authoritative model. At the time of writing, PUN allows a maximum of 20 concurrent users (CCU) in their free tier, with CCU above that value requiring a monthly subscription. Relatively low bandwidth limits are also imposed on a per-CCU basis, and these limits are not raised even in higher tiers of their payment model. In conVRged, we wanted to enable use cases beyond both the CCU and bandwidth limits, which led us to develop our custom networking solution, VelNet.

Before beginning work on a custom networking solution, we conducted a review of the existing solutions on the market. The primary motivation to switch away from PUN was to unlock the bandwidth and CCU limits that are imposed by all networking solutions that are hosted as a service, so the ability to self-host was a primary requirement. In the pursuit of open science and reproducibility, we also wanted to ensure that the networking solution was open-source, allowing research projects using the framework to be open-sourced without removing the networking layer. One common option that integrates well with Unity is Mirror, which is a high-level networking solution that is open-source and self-hostable. However, Mirror uses a client-host or dedicated server architecture, which did not fit with our goals for designing with high iteration speed and low cost.

5.1.3 Design of VelNet

VelNet is a custom networking solution that is designed to be self-hosted, open-source, and easy to use. VelNet is built to be very similar to Photon PUN 2, in order to ease the transition from projects that use that framework and have hit its limits or want the freedom of an open-

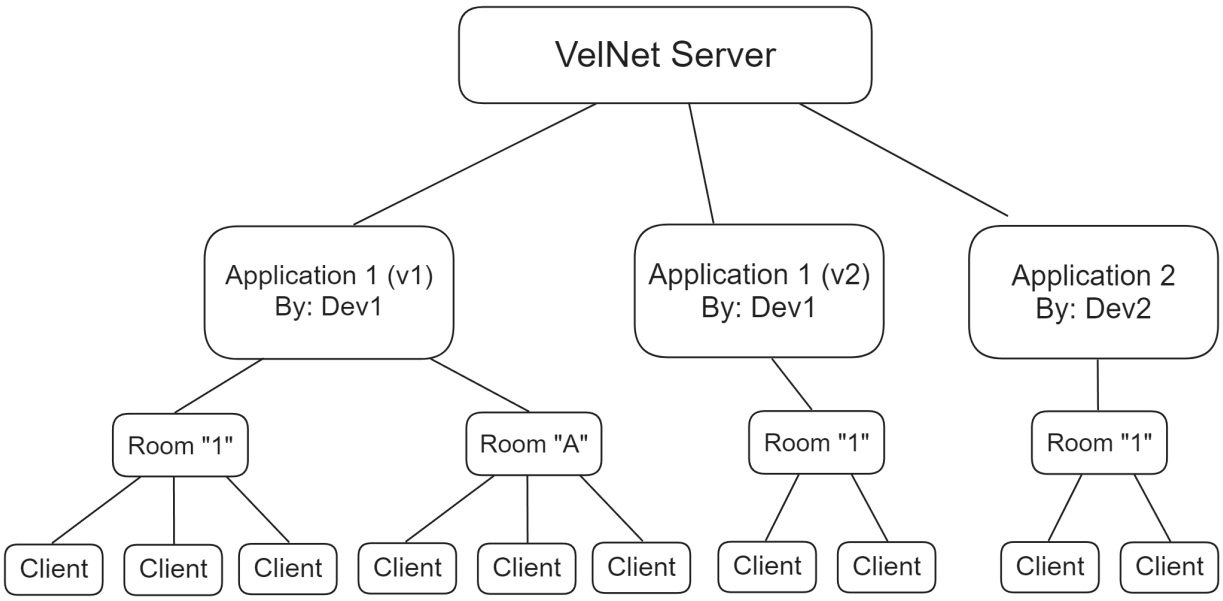

Figure 5.1: The architecture of the VelNet server and applications. Each application is independent without needing a second deployment of the VelNet server. Users in applications can connect to the rooms of the same name without colliding with rooms in different applications.

source solution that is not dependent on third party servers. There is no currently available networking software available for Unity with these features, so we believe this software to be a significant contribution to the space.

The server[120] is built in the Rust programming language[1] and can be deployed on Windows, macOS, or Linux on either x86 or ARM CPU architectures with a single docker command. If the available images available on Docker do not fit the target platform, the source is available and can be built for any platform supported by the Rust language. This is possible because of the lack of dependencies on third party networking libraries, as the communication is built using pure TCP/UDP connections. While this server application

[1] `https://www.rust-lang.org/`

does not act like a simple relay with no knowledge of the data being sent, it does not simulate the application state or require any knowledge of the design of the application being built. Updates to the client application do not require an update to the server. The server is also multi-tenant by default, so that many versions of the same game or different applications can be run at the same time using the same server application, as seen in Figure 5.1. Horizontal scaling is possible by running multiple instances of the server application, though a single application instance cannot span across multiple server instances. Considerable attention was spent on the ease of deployment of this server, with five different options for running the server code. The easiest is to run a single docker command that pulls the latest image from Docker Hub[2], but range to manual compilation of the open-source code using rust's compiler toolchain, cargo. In tests, the server was able to handle upwards of 70 connected clients, each continuously sending position updates without optimization, on a free Oracle Cloud Compute instance while using less than 20% of the CPU.

In order to facilitate the connections made to the server and handle the majority of the logic in this client-authoritative model, a Unity package must be developed. The Unity client is also available on GitHub[121], with documentation available at `https://docs.velnet.ugavel.com/`. The first step in any application's startup process when using VelNet is a connection to the server. This establishes whether the server is online and reachable, and tries to establish both a UDP and TCP socket connection. For the UDP connection, a small packet containing a user ID is sent to the server. Due to the unreliable nature of UDP, if no response is received within 100 ms, another packet is sent. This initial connection packet

[2] `https://hub.docker.com/r/velaboratory/velnet`

determines whether the client library denotes UDP as "connected" or not, and allows the system to make decisions about sending other data packets. The next step in the initialization process is a login step. When performed manually, this requires an app name and device ID. The app name allows the server to determine which other clients will be able to communicate with this client. Since clients on different versions of the same application should not be able to communicate with each other if their design has changed significantly, the default implementation of the Login function appends the version number of the application to the app name parameter of this packet. Some networking systems allow a separate "network compatibility version" to be maintained independently of the application version, such as Unreal Engine's (UE) "networkcompatibleversion"[124] or PUN 2's "Game Version". In Unreal Engine, the networkcompatibleversion is updated automatically when code changes are made, but not when only asset changes are made, and in PUN 2, Game Version is a string that is manually updated by the developer. In VelNet, the autologin process uses the global application version by default, but can be set to a manually updated value if needed. This way, application updates can be pushed to a subset of clients without breaking the ability for all clients to connect to each other. This is useful for minor fixes or for updates that only affect a single platform, increasing iteration speed, especially in situations where distribution on one or more platforms is time-consuming. After connecting, the client is given another ID by the server. A central problem in network library development is minimizing bandwidth usage, so the user-provided ID is substituted for a consecutively-generated integer as it is sent with many network packets to identify the intended target user.

The client library is designed to minimize the amount of work required for a simple prototype application. To this effect, options for automatic connection to the server, automatic login with the device ID of the client and server name determined from the project settings of the application are enabled by default. These settings can be seen in the "manager" component in Figure 5.2, which is the only component that needs to be added to the scene in order to set up the VelNet library. High-level components for syncing common types of objects are also supplied to serve common use-cases without the need for code or significant time investment. In research, starting new projects to test the feasibility of new ideas is more common than the commercial games and VR applications industry, which traditional networking libraries are built for. This heightens the need for a system designed for rapid prototyping. At the same time, research-focused applications often deal with unique requirements that make standardized components inadequate. To this end, VelNet includes access to low-level transmission functionality as a first-class customer. In the case that an application needs to send asymmetric or sporadic data, helper functions allow the user to send messages to all, some, or an individual user. The helper function adds required data such as the identification of the target recipient, message type, and instructions for the library to deliver the message to the correct instance of the script on the recipient's computer. Without this structure low level networking is still possible, but is made much more tedious.

An example of a minimal VelNet application is shown in the following code snippets. First, a manager class joins a room on login, then spawns a player prefab when the room is joined.

```
public class BasicVelNetMan : MonoBehaviour
{
```

```
    public GameObject playerPrefab;

    private void Start()
    {
        // join a hardcoded VelNet room as soon as we log into the server
        VelNetManager.OnLoggedIn += () =>
        {
            VelNetManager.JoinRoom("BasicExample");
        };
        // then once we join the room, spawn our player prefab on the network
        VelNetManager.OnJoinedRoom += _ =>
        {
            VelNetManager.NetworkInstantiate(playerPrefab.name);
        };
    }
}
```

The second snippet is a script on the player object, allowing basic movement if the object is a local player. To synchronize player position, the default SyncTransform script can be added to the player object.

```
public class PlayerController : MonoBehaviour
{
    public NetworkObject networkObject;

    private void Update()
    {
        if (networkObject.IsMine)
        {
            Vector3 movement = new Vector3();
            movement.x += Input.GetAxis("Horizontal");
            movement.y += Input.GetAxis("Vertical");
            movement.z = 0;
            transform.Translate(movement * Time.deltaTime);
        }
    }
}
```

Outside of custom binary messages, a NetworkObject component is needed to send data across the network. NetworkObjects handle the concept of ownership, which defines which instances of objects are responsible for sending data and which are receiving. In a networked system, both the local user and the remote user have a copy of the scene hierarchy. Corresponding objects in the two hierarchies are identified by having the same network ID. Once two remote objects are linked by their network IDs, they are able to communicate with each other, but rather than establishing a new connection to the server for each object, they communicate through the central VelNet manager. Many monolithic scene objects developed in applications consist of several common parts that would benefit from pre-established syncing behaviors. To allow this re-use of existing components within larger structures, NetworkComponents can be attached to NetworkObjects to send messages through the object. Again, to prevent the duplication of network IDs and the overhead that corresponds with the management of ownership that a new NetworkObject entails, the NetworkComponents are given an ID that allows the NetworkObject to multiplex the packets to the correct sub-component.

To create a minimal custom synced object with arbitrary binary data, the following script can be used. Any number of custom data types can be serialized beyond the Vector3 shown here, and the abstract SyncState class will handle periodic the transfer of data between clients with network optimizations when the contents of the data do not change.

```
public class SyncMyObject : SyncState
{
    protected override void SendState(BinaryWriter binaryWriter)
    {
        binaryWriter.Write(transform.position);
    }
```

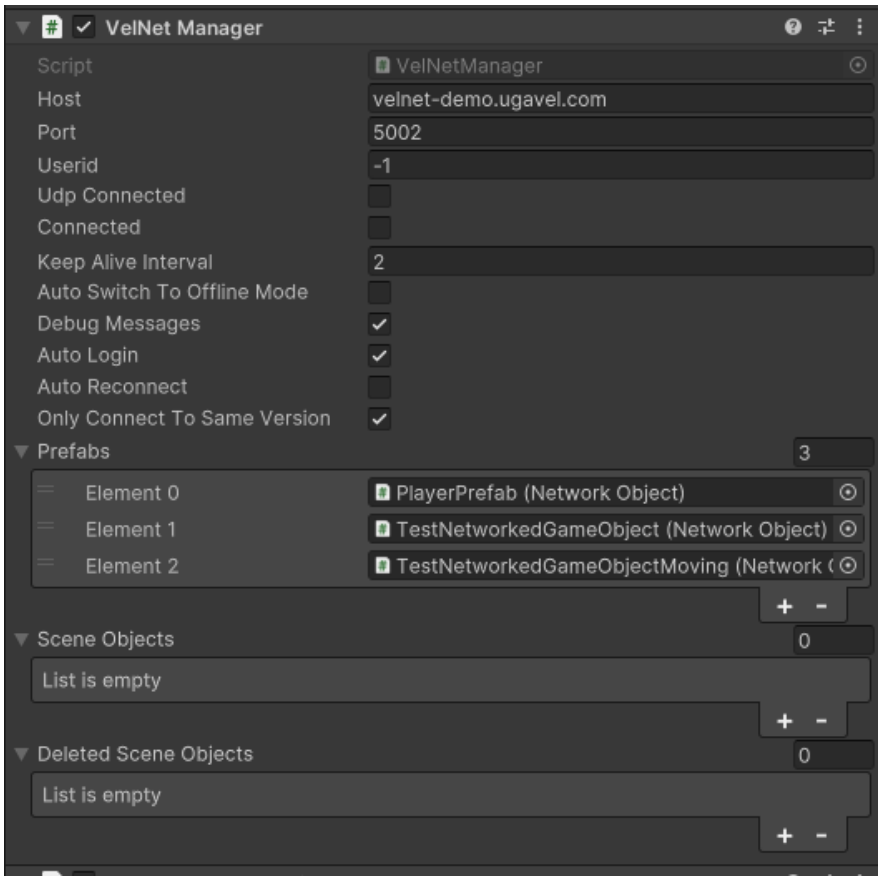

Figure 5.2: The VelNet Manager component in Unity. This component is responsible for managing the connection to the server, and is the primary interface for the client application to interact with the VelNet library.

```
    protected override void ReceiveState(BinaryReader binaryReader)
    {
        transform.position = binaryReader.ReadVector3();
    }
}
```

Undo

The ability to undo actions performed in an application is often a desirable feature but is not often implemented in research applications[97]. Undo provides significant value to the

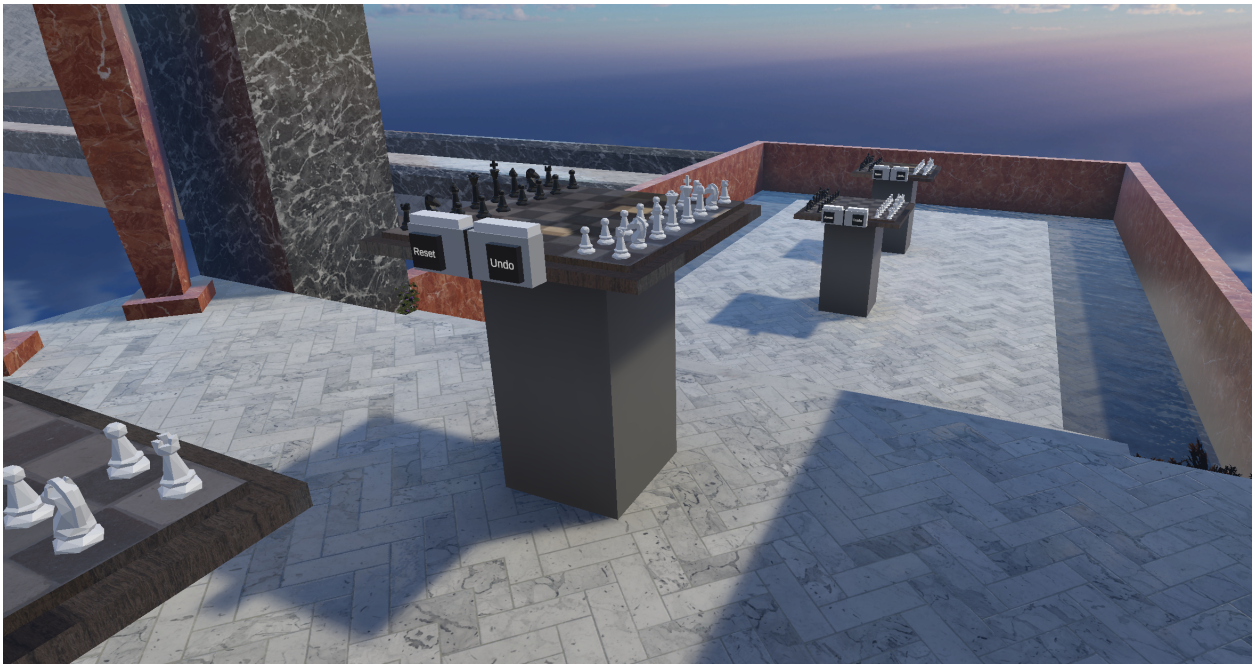

Figure 5.3: One of the environments in *conVRged* showing chess boards with an undo button. The undo feature leverages the VelNet library to record the network state of the pieces on the board.

user experience[16], but can be a complex addition to an application due to the need to record every action by the user in a reversible way. It is also possible to record the full state of the environment at periodic or hand-picked intervals, but this approach quickly becomes space-prohibitive with larger projects. This need to record the changes to the virtual world over time coincidentally aligns with the needs of a networking system. When building VelNet, we recognized this similarity and built a component that records the network packets as they are sent, then allows the user to revert to a historical state through the use of an undo feature. In order to align with the needs of any specific research application, this system was not built to be global; rather, an UndoGroup component can be added to an object that has

a list of other components assigned to it to record. Virtual worlds can be large, with many parts acting independently. This system can undo a specific object's data without affecting the rest of the world. For example, in conVRged the chess boards have an undo button that allows each of them to be reverted to a previous state without affecting the other boards or even other actions by that user in the same time period. While the undo system is built into and shipped with the VelNet Unity library, we believe it to be an ideal case study on the creation of flexible tools using the VelNet library, as it was built without having to modify the rest of the VelNet source code and could live in the user-space code outside of the package if it was built for a specific research application's needs.

Test project and examples

Ease of learning is critical to adoption of a software package, so care was taken in providing simple as well as detailed examples. Four sample scenes ranging from the most basic possible scene that can connect and send data, to a fully-featured example that demonstrates spawning and syncing more complex objects as well as player avatars and more complex connection scenarios. These examples are distributed using the Unity Package Manager's sample system, through which they can be imported in user-space code, so they can be modified for experimentation. The samples were also useful when testing the performance and capabilities of VelNet. While the benefit of test-driven-development (TDD) is debated in a situation and at a scale such as this[50], adding tests has been found to significantly reduce the prevalence of bugs and unexpected behavior[5, 81, 49]. The samples were designed to span a range of expected scenarios faced by the library, and often helped to identify issues in

a simpler environment than the larger conVRged application. One of the tests also included the ability to spawn a large number of objects at a time, allowing us to determine that the server and client setups were sufficient to support higher loads than we would expect in an application such as conVRged. In one such test, we had several thousand shared objects being synced across tens of connected clients without a noticeable slowdown or significant load on the server. This gave us confidence that the limitations for large-scale networking would remain in the client connection quality and client rendering and simulation performance when dealing with large numbers of objects.

5.1.4 Low network availability, locally-hosted, and offline situations

It is often necessary in research environments for headsets to be deployed in environments with non-existent or inadequate internet connections. Whether this is because of security restrictions of the local network or the lack of connectivity, it was important that applications built with VelNet did not cease to function entirely without an internet connection. The first fallback that needed to be developed was an offline mode. Due to the way VelNet and most networking libraries are built, significant parts of application logic are dependent on callbacks from the networking system. For an application to support an offline mode by itself would require either duplicating this functionality or simulating a networking connection itself and calling the functions manually. To avoid this, VelNet detects a missing network connection and has a default option to enter an offline mode. VelNet then sends the appropriate "packets" to the rest of the local software to simulate the standard connection procedures and handshake. When messages are sent to the server by application code, automatic responses are generated

locally by a virtual server with the assumption that the local user is in a room by themselves. Thus, without checking the global "offline mode" flag, the lack of a network connection is completely opaque to the user-space code.

Another common situation, especially with VR headsets, is frequent disconnection and reconnection events from the network. When the Quest headset is not worn, the screen turns off after a short timeout, then if the headset is not plugged in, the Wi-Fi antenna is turned off. In research study environments, we found this situation to be common, and it was critical that the headsets would reconnect to the same VelNet room automatically after disconnecting. We built a system in VelNet that detects when a connection has been broken, then periodically tries to reconnect. In order to avoid common errors in user-space code that could occur if the assumption is made that each connected user in a room has been available to receive all the messages they have sent, the reconnected client first leaves any rooms they were connected to, then reconnects to the same room. This way the reconnected user is treated as a new user, and any logic that was developed to set up their presence in the scene and synchronize state can reoccur.

The third solution to low availability of internet is to host the server on the local network with either a self-provided access point or on another device on an existing local network. In order to facilitate this connection, an API was added to VelNet to allow the changing of the connection host at runtime. This is generally not needed in this style of networking library, but it is particularly useful for research or other on-premise applications. Hosting the server software locally has several other advantages beyond the lack of reliance on the global internet. Due to the physical proximity of the server to the clients, latency is significantly reduced,

which is particularly important in VR. Bandwidth to a server is often limited not by the local network speed, but rather by the bandwidth allocated by the Internet Service Provider (ISP), and using a server on the local network removes that second limitation. Hosting locally can also comply with security restrictions in some corporate environments, where transmission of user data to a server on the global internet would require an approval process to ensure that the data is not being stored or re-transmitted outside the control of the administrator. For example, on one project for which we deployed conVRged, the local network on location was not available to us, but we were able to provide local network access with a consumer router and PC that was running the VelNet server locally. Other research-oriented services were all running on that local PC, allowing us to avoid transmitting private user data across the internet, as well as providing a much higher bandwidth connection for research data even had there been a global internet connection available. Because the router we deployed was located centrally to the VR experience rather than in a position to equally serve the occupants of the building, we were able to provide a more stable connection to the approximately 10 headsets that were simultaneously connected and operating.

Voice communication - VelVoice

While not critical in VR applications, voice communication is ubiquitous in immersive applications, allowing the virtual avatars and communication to mimic communication in the real world[128, 69, 6]. Prior to developing our own solution, conVRged used Dissonance voice chat[27], a proprietary asset that handles the conversion of microphone data to a binary signal with attenuation, background noise removal, and echo cancellation. This binary stream of

data was then sent across the standard VelNet channels and distributed to other clients. The clients are then responsible for playing back the mono-channel audio at the correct location in a spatialized way. Due to the proprietary nature of the Dissonance asset, it (and any software written to support it) could not be included in the standard VelNet package, as every user would need to purchase the asset to use VelNet. Instead, a secondary package was created that included only the code necessary to interface VelNet with Dissonance[119].

In an attempt to reduce the number of proprietary dependencies that prevented us from distributing the raw source of conVRged and future projects, VelVoice was developed as an alternative to Dissonance voice chat. VelVoice uses the open-source Opus codec to encode and decode microphone and audio streams[115, 62]. VelVoice does not have the same echo cancellation and background noise removal features as Dissonance, but we were encouraged to find that in the Meta Quest family of headsets, audio quality was still high quality. We encourage other VR developers to have the confidence to build their own audio solutions, as most existing audio communications solutions are proprietary and limit the proliferation of open software and open science.

5.2 Persistence - VEL-Connect

VelNet is a realtime communication product that facilitates rapid synchronization of game state, but it does not store long-term state or user data. When all users of a particular room have left, no memory of the state of that room is kept, as the clients were the only source of that state information. Many applications have a need for persistence of both the state of the

environment as well as customization of the users themselves through nametags or avatars. One solution to this problem is to continue the model of VelNet, where devices are store the state in memory, then persist that state to disk when they disconnect. For personal headsets this works well, but in research applications there is often a fleet of headsets that is deployed, and even if the same user enters the environment for a second time, there is no guarantee they will be using the same device. To solve these problems, a networked persistence layer is needed to maintain long-term data across devices.

At a high level, VEL-Connect[117] acts as a persistent storage counterpart to VelNet. It can be used as a key-value store as a networked replacement for Unity's PlayerPrefs system, to save the state of a virtual environment across sessions, to share a profile of user data across multiple devices, or to easily switch between different user profiles on a single device. Just as with VelNet, particular care was taken in the ease of self-hosting and client setup for rapid prototyping. A documentation website is available at `https://docs.velconnect.ugavel.com/`, which provides instructions for basic setup and usage of both the client and server. Again as with VelNet, VEL-Connect is multi-tennant, meaning multiple independent applications can use the server without interfering with each other's data or requiring any initialization or authorization steps to bootstrap a new application.

Though VEL-Connect was first built from the ground up using a custom API surface, the current version makes use of a customized version of the open-source Pocketbase[3] framework. The switch to Pocketbase primarily provided advantages in the automatic generation of JavaScript client SDK code that made the development of web dashboards easier to maintain,

[3] `https://pocketbase.io/`

as well as an authentication system. At it's core, Pocketbase is a sqlite database that can be customized to fit the target application [1]. VEL-Connect defines a set of tables that enable the intended use-cases of use profile data, room data, and object persistence, as seen in Figure 5.4. Profile data is achieved through an account system that allows a user to transfer their profile data across multiple devices. Usually this is achieved in consumer electronics through a login system, where the user enters a username and password on the device to authenticate and retrieve their profile data. However, in VR the usability of keyboards is significantly reduced, leading many to develop and evaluate alternative methods of text entry[54, 28, 41, 63]. To mitigate this issue, we chose to develop a pairing system that causes the typing to occur only on a companion web app and not the headset itself. When the application in VR is started, the device registers itself with the VEL-Connect server and shows a four digit pairing code to the user. The user can log into the web dashboard with a traditional username and password, then enter this pairing code in the dashboard, which assigns their profile data to the current device. This workflow solves the more traditional one-to-many case where one user could log into many potential devices and needs to bring their user data with them. However, more uniquely in a research setting, we also needed to solve the many-to-one case where one user needs to manage the profile assignments of many users. Since we still needed to support the original one-to-many case, the problem became a many-to-many mapping between user profiles and devices. For example, a researcher conducting a user study in a lab for a multi-user experience with standalone VR headsets needs to be able to assign the correct user profile to each of the headsets remotely without needing to log in and out of the dashboard multiple times.

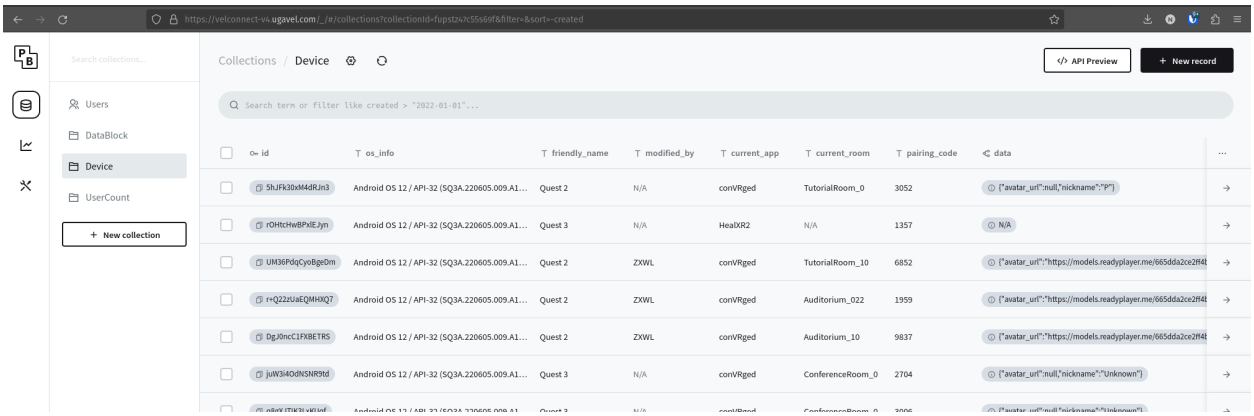

Figure 5.4: The Pocketbase VEL-Connect dashboard showing a list of known devices with their attached profile data.

All of these scenarios are facilitated by the VEL-Connect dashboard in conVRged. As seen in Figure 5.5, once a device is paired to an account, it shows up in a list of available devices. Many individual profiles can be created, then each profile can be assigned to one or many of the devices using a dropdown. These profiles contain arbitrary JSON data that can be defined by the application, but these data are intended to represent the state of the user, such as avatar customizations, nametag, or even a study ID.

The second major use for VEL-Connect is to persist room state changes. The VEL-Connect handles the synchronization of the server's key-value store representing the room state to the local client, and it is up to the application code to apply this data to the scene. For example in some scenes in conVRged, the user can choose to enable "night mode" to change the environment for a stargazing experience using a virtual telescope. Because this configuration parameter is defined in human-readable format in the VEL-Connect database, the conVRged dashboard can show a checkbox for enabling this setting permanently before

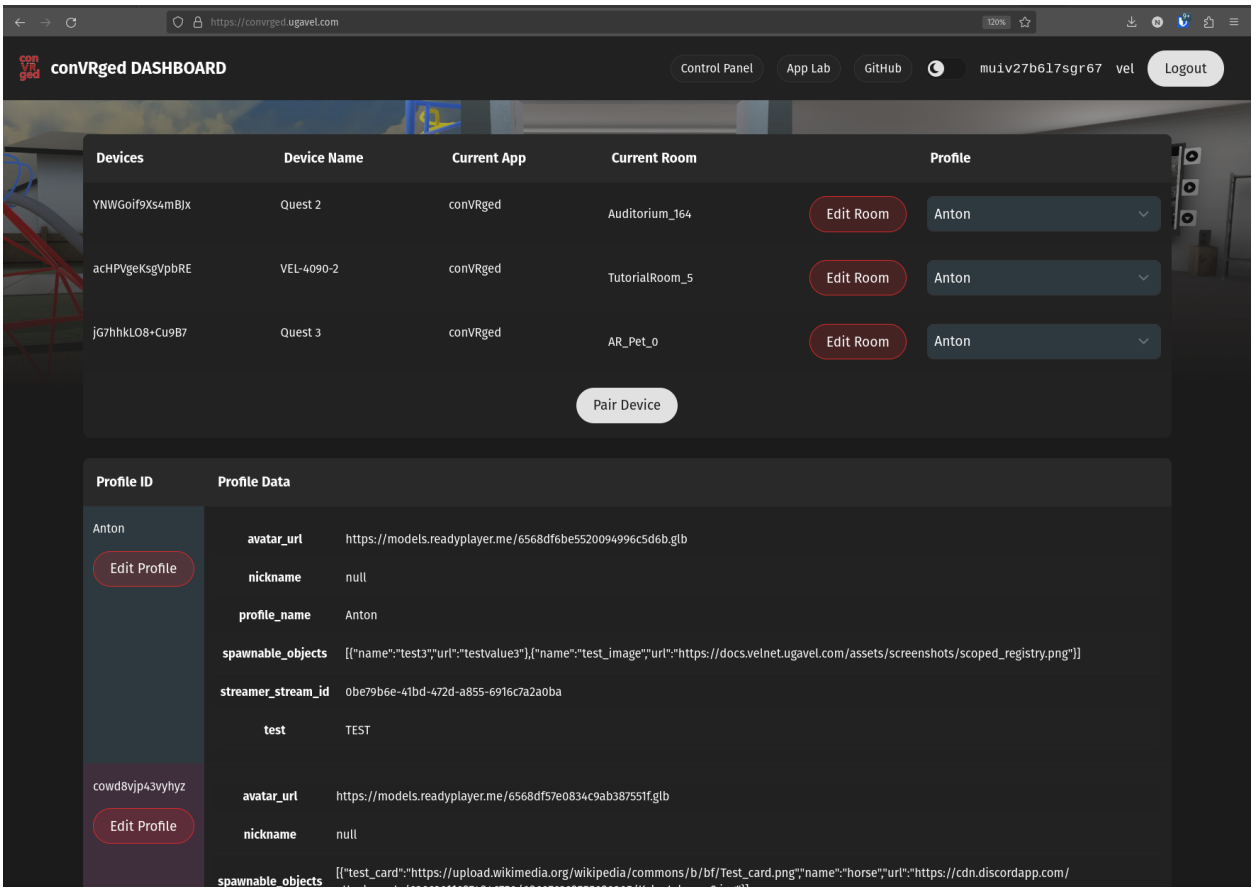

Figure 5.5: The homepage of the conVRged VEL-Connect dashboard with several paired devices.

any headset has even entered that room. We believe this feature has great utility for pre-configuring study spaces, especially if several variations need to be configured with complex combinations of parameters. The state of the day/night switch in this example could also have been synchronized with VelNet and even persisted if necessary by an automatic serialization of the state, but the binary representation that VelNet uses for rapid real-time communication is not ideal for consumption and modification by an external dashboard or study management software. Therefore, use of this method of persistence is ideal for data that does not rapidly change or configuration that represents input parameters for a room that should be configured externally to the VR application.

The third method of persistence provided by VEL-Connect is the automatic serialization of VelNet data. A utility component can be added to any VelNet NetworkObject that periodically uploads the object's state to the VEL-Connect database and restores it to the local client as long as the client is the first to enter a VelNet room, meaning there is no more up-to-date value for the state of the object. This works well for interactable objects that can be moved throughout the environment, whose primary synchronized state is a position and rotation. This method of persistence also works for spawned objects. While in a VelNet room, users have the option to spawn new synchronized objects. Usually these objects would despawn as the last user leaves the room, but such objects often play a large role in the customization of a room, such as placing posters or building structures.

A common problem when attempting to do productive work in a VR headset is how to upload relevant documents or other content to the headset. VEL-Connect allows users to upload documents to the web dashboard from their computer where the documents originate,

then spawn those documents inside the VR application. All of these features serve to reduce the amount of configuration and text entry in the headset, allowing the user to use the headsets for their core strengths.

5.3 Screen sharing - VEL-Share

Another approach to introducing content relevant for productive work to the virtual environment in a VR headset is to stream a separate device's screen. Now a common feature in consumer headset software such as Apple's Vision OS[104], Meta Horizon Workrooms[113], or commercial 3rd party software such as Immersed[4] or Virtual Desktop[5], the benefits of using existing non-VR devices in a virtual environment have been well-explored [82, 2]. What these existing solutions do not offer, however, is the option to easily share screens into a custom research applications built to run on a variety of platforms.

VEL-Share is a custom screen sharing solution that allows users to share their screen to a VR headset. The system is built around the concept of a Selective Forwarding Unit (SFU) to enable the distribution of video data from one client to an arbitrary number of viewing clients. SFUs work by ingesting data from one client, then allowing other clients to query the server for the data output stream. SFUs effectively allow a single client to upload a data stream to a large number of receiving clients without requiring more than a single upload-stream's bandwidth. VEL-Share's SFU does not re-encode the media stream, so the CPU and GPU performance requirements of the server are very low. A similar architecture is common in

[4] `https://immersed.com/`
[5] `https://www.vrdesktop.net/`

group video conferencing applications, though sometimes a Multipoint Conferencing Unit (MCU) is used instead to re-encode the media streams to reduce downstream bandwidth. The VEL-Share server is deployed using Ion SFU[6], and screen-sharing clients can send data using a custom dashboard built with an open-source JavaScript library[7]. The conVRged dashboard includes a GUI to enable screen sharing on any platform that supports browser-based screen capture, as seen in Figure 5.6. This level of accessibility is greatly beneficial, as it reduces the friction for a new user to share an arbitrary document to the virtual world.

While the web-based screen-sharing tool has several controls for resolution and encoding format, we experienced issues with some browsers' ability to share desktop audio. Audio sharing is not necessary in all situations, but to complete the user experience, a desktop application was built to perform the same sharing as the web-based tool but without the limitations imposed by browsers on sharing audio and DRM-protected video content. This application can be seen in Figure 5.7.

The receiving software uses a WebRTC stream to directly load the video stream from the SFU into a texture in the virtual world. Earlier implementations of VEL-Share used a browser abstraction to load a web page that loaded the stream. While this was flexible and allowed for a single client that could be used inside and outside of VR, the overhead of the browser's engine was detrimental to application performance, which led to the development of the direct WebRTC approach. Adding the stream viewer to a Unity application is straightforward, requiring only the import of a Unity package[118] and the addition of a single object to the scene. As the development of software designed to test a research idea evolves, the

[6] `https://github.com/ionorg/ion-sfu`
[7] `https://github.com/ionorg/ion-sdk-js`

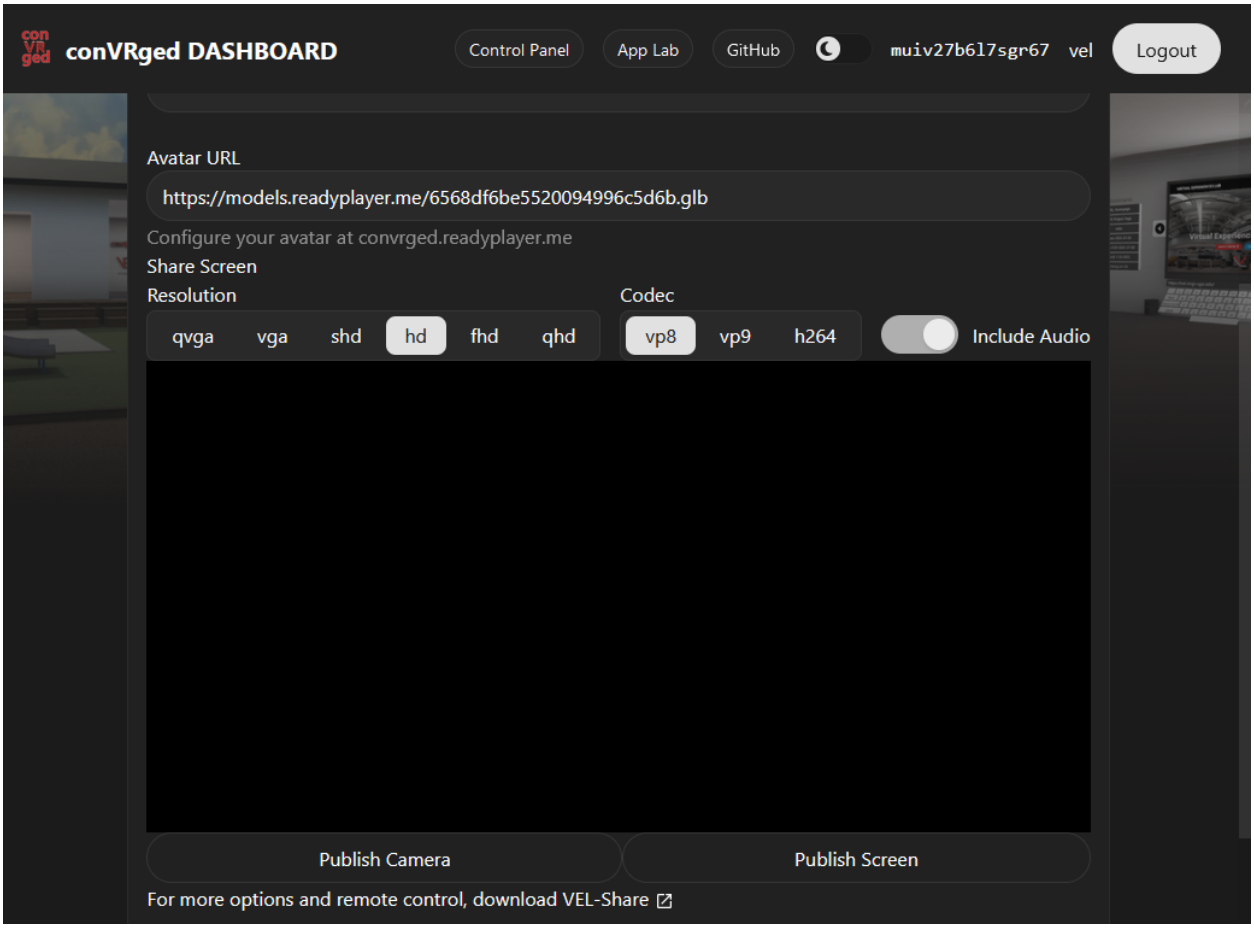

Figure 5.6: The conVRged dashboard with the profile editing page open. This page contains the screen-sharing widget, which sends the user's screen to VEL-Share via the browser. This page also handles the addition of the screen share's stream ID to the user's profile so that it can be accessed in VR.

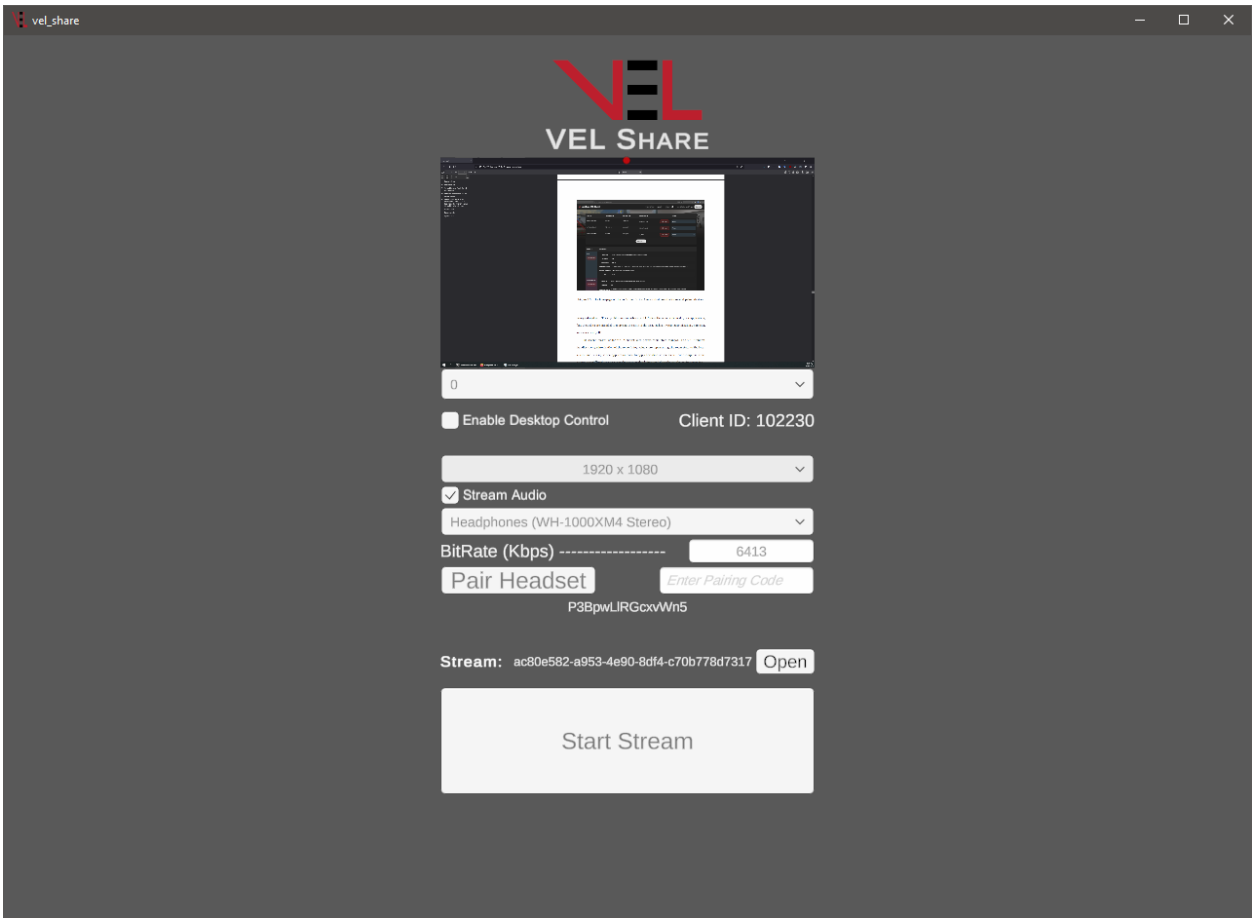

Figure 5.7: The VEL-Share desktop application. Just like the web-based tool, this application allows the user to share their screen to the virtual environment, but through a native application not contstrained by browser-based APIs.

requirements often change significantly, necessitating the switch to a different VR headset, possibly one that runs on a different platform architecture. By designing the receiver software using low-level existing web standard components, there is a greater likelihood that the new headset or game engine will be able to support the reception of this data without an additional large development effort. Just as with the other components detailed in this chapter, the goal of creating flexible and useful components has allowed these components to be reused in several projects in our laboratory.

One difficulty in designing a screen-sharing system that works with a server that supports multi-tenancy is the need to determine which clients receive the video stream. To achieve this, we were able to make use of one of the other modular components described in this chapter, VEL-Connect. On the web-based sharing interface, the conVRged dashboard requires a login, which allows users to assign profiles to devices that they have paired. By putting the screen sharing widget on the page for a specific profile's data, the device will be updated with an ID of the stream. Once a device in VR receives the updated profile information and determines that the stream is available, it can distribute the information about the stream ID to the other clients in shared virtual space via the VelNet real-time networking system. With this approach, each user can have a screen shared to the SFU, and the application can choose which user's stream to show to the virtual room through a custom selection interface. For example, a group of users could be using conVRged to share individual presentations sequentially. Each presenting user would share their screen through the dashboard before they enter VR, then the virtual projector screen could switch to the next presenter's screen by virtue of the physical presence of the user at the front of the room or by pressing a single

button. When using these kinds of screen-sharing mechanisms, we have found that friction often makes the experience slower or more cumbersome than sharing outside of VR, and we believe that the ability to easily share screens in a VR environment is a significant step towards making VR a more productive environment for research and work.

5.4 Connecting infrastructure - VelUtils

Unity is a general-purpose game engine designed to accommodate a wide variety of applications, so the development of a more specific category such as VR applications narrows that focus and requires a common set of tools or utilities. Even further, any VR research application requires instrumentation and logging tools that are not built into the engine itself. This problem of the sometimes significant amount of effort required to bootstrap a project is well known in the industry, which has led to the development of the Unity Asset Store, in which developers can download pre-built assets and toolkits. One such asset is VRTK, self-described as "a collection of useful scripts and concepts to aid building VR solutions rapidly and easily[127, 21]". Many other such projects aim to achieve similar goals, however inevitably each toolkit is designed to solve the problems of the developers who built it or to match the expectations of the industry[26, 93, 74]. In VR research projects, the goal is often to redefine the standard interaction paradigms making the use of pre-built tools often more cumbersome that building the components from scratch. We recommend in many situations that VR researchers consider building components central to their research question from scratch or lower level components to avoid the opinionated design of toolkits and the overhead

of learning the toolkits rather than the base level tools of the engine. However, we do believe there is great value in developing an internal toolset that makes use of the same modular design principles as off-the-shelf toolkits.

During the development of conVRged, we accumulated such tools through the natural development of individual components, adding them to a shared repository called VelUtils[123]. The modules in this package contain features such as interaction scripts for 3D selection and manipulation tasks, controller input abstractions, locomotion scripts for common forms of movement in VR, persistent configuration utilities, and a set of utilities for logging research data[122]. Many of these components serve a similar purpose to those found in commercial toolkit offerings, but generally have a smaller level of abstraction and are designed for smaller scale projects as found in the HCI research space. One component in particular is highly relevant to the research space and is not common in commercial toolkits: a data logging system designed to be self-hosted.

The logging system in VelUtils allows researchers to define a set of headers for a tabular data file, then use a single line of code to add a new line of values to the output. To reduce repetition, a customizable set of common headers such as timestamp, a unique study identifier, or study condition can be added to every row. Commercial logging systems exist, but they are generally focused on a debugging and user analytics perspective[105, 19, 66]. VelUtils's logging system is designed to output data in a format that is useful to VR researchers, with comma separated value (CSV) files that are generated locally. Optionally, the data can be sent to a server during or after the VR experience with an in-application consent mechanism using the open-source server software. The ability to export data directly from the headset

to a private server has proven to be useful the approval process deployment to VR app stores such as Meta's App Lab, and having the study user data readily available in the format that is most useful for analysis has been a significant time saver in the analysis phase of our research projects. By using these same logging scripts across multiple research projects in our laboratory, we have been able to reduce the duplication of effort in the writing of analysis scripts as well as the development of the logging itself.

5.5 Packages, Modularity, and Installation

All of the modular Unity assets described above were developed to make use of Unity's Package Management system[8]. Furthermore, the packages were uploaded to a custom NPM registry at `https://npm.ugavel.com/`. Once the registry URL has been added to a project, each of the packages in the registry can be added to the project with a single click. One of the key benefits of this system over installing packages directly via a GitHub URL is the ability to install specific historical versions. This versioning has proved useful in our laboratory projects, as breaking changes to a specific package could be added without requiring every other project to update to the latest version until its own dependencies were updated to support the new version. By using the Unity package manager system, a strict form of dependency isolation is enforced at the compiler level. These restrictions made reusability of components easier to achieve and minimized the amount of work required to add a new package to an existing larger project.

[8] `https://docs.unity3d.com/Manual/upm-ui.html`

Chapter 6

Conclusion

In this dissertation, we have described a system that has been developed for, validated against, and used for evaluations in a variety of application areas. Through the work of several published research projects, we explore the value of individual components that have been developed for the single cohesive application *conVRged*.

First, a novel immersive engineering education application was developed and evaluated. 85 participants were involved in a user study that consisted of the use of a land surveying exercise that mirrors that done with real equipment in an introductory engineering course. Participants were assigned to one of two conditions, either working individually or collaboratively in a pair with another participant, though all were guided by the experimenters, who were also immersed in the same virtual environment. Results from this study found high levels of enthusiasm and engagement with this teaching approach. Significant gender disparities were found on several metrics, including task load and presence, even after controlling for video game experience. These gender disparities match previous work in the VR

space[40, 68]. While important to consider, the differences found here do not necessarily represent an increase in these task load indicators over non-VR activities for traditionally underrepresented engineering populations. Promising avenues for successful implementation in the engineering curriculum were found from this work.

Spawned from the desire to create a shared spatial workspace environment, another project was completed that evaluated the use of a point cloud annotation tool. This work involved the creation of an immersive point cloud visualization tool with a continuous level of detail system to smoothly increase detail at the center of the viewport and introduced a novel technique that increased detail in the task area by taking into account the position of the user's tracked hand positions. The tool also enabled users to annotate individual points in large point clouds at interactive framerates using a painting metaphor. As a comparison condition, an analogous non-immersive desktop tool was created with the same visualization techniques but using a mouse to select points. A novel distance selection technique was used to sample the depth of the points being selected and avoid selecting points in the background. Results from this study found significant increases in both user performance and user satisfaction in the immersive tool, and a nuanced analysis of task performance found advantages of both interfaces in different aspects of the task. We recommend a hybrid approach for the development of such a tool for productive applications in the future. The point cloud visualization tools were incorporated into *conVRged*, as seen in Figure 6.1.

A third user study was developed to evaluate the locomotion systems used in *conVRged*, with the addition of a comparison to hand-tracked input mechanisms. *conVRged* uses a hybrid approach of ray teleportation, snap-turn, and grab-move. This work compared these systems

Figure 6.1: A presentation space in *conVRged*, including a point cloud representation of a LIDAR scan of a cotton plant.

across hand-tracked and controller-tracked conditions by performing a user study with three separate tasks designed to span the spectrum from small-scale stationary activities that we expected would favor the grab-move technique to a long-range task that was only possible with the teleportation mechanic. In our results, we found confirmation of our predictions that some tasks were more optimized for different locomotion techniques, and we found benefit in the use of both systems simultaneously for a mixed search and manipulation task. We also found and quantified significant preference and performance advantages for the controller-tracked condition over the hand-tracked condition, which matches prior work in the area, and is possibly influenced by inherent latency differences between the technologies.

Finally, we present a detailed analysis of the design and development of the individual components that were constructed throughout the development of the previous projects. In some cases, these components represent novel solutions to the problems faced in this space, though in all cases they serve to improve the cohesive real-world application *conVRged*. Each of these components has been used in other projects outside the scope of this dissertation, proving their reusability and modularity.

Many of the components and tools discussed in this dissertation are open source:

- **conVRged**
  - `https://github.com/velaboratory/conVRged-OSS`
- **DataFoldvr** - Point cloud annotation tool
  - `https://github.com/velaboratory/DataFoldvr-Virtual-Reality-Point-Cloud-Annotation`

- **VelNet** - Self-hosted networking system similar to Photon PUN
  - `https://github.com/velaboratory/VelNetUnity`
  - `https://github.com/velaboratory/VelNetServer`
- **VEL-Connect** - Out-of-headset configuration dashboards for persistent data
  - `https://github.com/velaboratory/VEL-Connect`
- **VelUtils** - Locomotion and interaction toolkit built for simple setup
  - `https://github.com/velaboratory/VelUtils`
  - `https://github.com/velaboratory/VelUtils-OVR`
- **VEL-Share** - Efficient screen-sharing using WebRTC
  - `https://github.com/velaboratory/VEL-Share-Unity`

6.1 Other applications and future directions

This work distinguishes itself not only by the development of a set of reusable and well-designed tools but by the testing of these tools through the co-development of a versatile and battle-tested social XR application deployed for real-world use cases. The application *conVRged* has been used for several projects beyond those described here. Examples of these projects include:

Virtual Fitness Buddies (VFB) - *conVRged* was used to rapidly deploy a shared AR environment, incorporating components from a previous project and adding features

such as Meta's Spatial Anchors to enable a collaborative AR space. A pilot study for this project is currently ongoing, which is encouraging kids to stay active by playing with virtual basketballs and interacting with a virtual pet that reflects the exercise they have been doing, tracked by a fitness watch. By leveraging *conVRged* for this project, kids can play in the same environment, seeing each other's pets and throwing balls to each other.

Periodic lab demos - *conVRged* contains the work of many individual projects, so it serves as an ideal space for demonstrations of our work. Because of its multiuser nature, demos can scale to larger groups, and a desktop computer monitor can join the same environment to show a static view of the environment with virtual users engaging.

Productive lab meetings - During the COVID pandemic, our lab meetings were conducted in *conVRged*, with features such as screen sharing enabling group presentations and engaging conversations. One such meeting space can be seen in Figure 6.2.

Chess and Debate Club - *conVRged* has been deployed for use in a local after-school chess and debate club for middle and high school children. A dedicated environment for this combination of activities was created with a paired presentation space and competitive chess spaces.

Mind Mapping - *conVRged* has been used for a study in the School of Journalism to evaluate collaborative use of a spatial mind mapping application. Nodes and connections can be added and placed in 3D space, each with a choice of colors and labels.

Beyond the direct uses of *conVRged*, the components built for *conVRged* have also been used in separate projects:

Figure 6.2: A versatile meeting space in *conVRged*. Due to spatial audio, users can talk in small groups without interrupting other conversations.

Teaching - Kyle Johnsen's undergraduate and graduate Virtual Reality class has used components such as VelNet and VelUtils through the last few years. Kyle has reported student success with these tools. Each semester, student groups create a wide variety of applications using these tools.

HealXR - A research project called HealXR intends to evaluate methods to reduce phantom limb pain in amputees through the use of XR. This project makes extensive use of VelNet, VelUtils, VEL-Share, and VEL-Connect to achieve this goal. Feedback from this project has helped improve the robustness of the tools.

ENVISION - An XR tool to evaluate writing and drawing in VR, ENVISION uses VelNet and VelUtils to support collaborative drawing experiences.

Robot Training - An educational tool for teaching students to program a UR10 collaborative robot has been built using VelNet for networking and VelUtils for object interaction. A version of this tool uses a shared passthrough XR environment to enable manipulation of a virtual robot while remaining engaged in the real world.

Bibliography

[1] Grant Allen et al. "Introducing SQLite". In: *The Definitive Guide to SQLite* (2010), pp. 1–16.

[2] Konstantinos C Apostolakis, George Margetis, and Constantine Stephanidis. "'Bring Your Own Device'in VR: Intuitive Second-Screen Experiences in VR Isolation". In: *HCI International 2020–Late Breaking Posters: 22nd International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22*. Springer. 2020, pp. 137–144.

[3] Felipe Bacim, Mahdi Nabiyouni, and Doug A Bowman. "Slice-n-Swipe: A free-hand gesture user interface for 3D point cloud annotation". In: *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE. 2014, pp. 185–186.

[4] Jeremy N Bailenson et al. "The independent and interactive effects of embodied-agent appearance and behavior on self-report, cognitive, and behavioral markers of copresence in immersive virtual environments". In: *Presence: Teleoperators & Virtual Environments* 14.4 (2005), pp. 379–393.

[5] Thirumalesh Bhat and Nachiappan Nagappan. "Evaluating the efficacy of test-driven development: industrial case studies". In: *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. 2006, pp. 356–363.

[6] Frank Biocca and Mark R Levy. *Communication in the age of virtual reality*. Routledge, 2013.

[7] Chuck Blanchard et al. "Reality built for two: a virtual reality tool". In: *Proceedings of the 1990 symposium on Interactive 3D graphics*. 1990, pp. 35–36.

[8] Simone Borsci et al. "Assessing user satisfaction in the era of user experience: Comparison of the SUS, UMUX, and UMUX-LITE as a function of product experience". In: *International journal of human-computer interaction* 31.8 (2015), pp. 484–495.

[9] John J Bosley. "Creating a short usability metric for user experience (UMUX) scale". In: *Interacting with Computers* 25.4 (2013), pp. 317–319.

[10] Doug A Bowman and Ryan P McMahan. "Virtual reality: how much immersion is enough?" In: *Computer* 40.7 (2007), pp. 36–43.

[11] Doug A Bowman, Ryan P McMahan, and Eric D Ragan. "Questioning naturalism in 3D user interfaces". In: *Communications of the ACM* 55.9 (2012), pp. 78–88.

[12] Evren Bozgeyikli et al. "Point & teleport locomotion technique for virtual reality". In: *Proceedings of the 2016 annual symposium on computer-human interaction in play*. 2016, pp. 205–216.

[13] Davide Calandra et al. "Arm Swinging vs Treadmill: A Comparison Between Two Techniques for Locomotion in Virtual Reality." In: *Eurographics (Short Papers)*. 2018, pp. 53–56.

[14] Nicola Capece et al. "A Preliminary Investigation on a Multimodal Controller and Freehand Based Interaction in Virtual Reality". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2021, pp. 53–65. DOI: 10.1007/978-3-030-87595-4_5.

[15] Christer Carlsson and Olof Hagsand. "DIVE A multi-user virtual reality system". In: *Proceedings of IEEE virtual reality annual international symposium*. IEEE. 1993, pp. 394–400.

[16] Aaron G Cass, Chris ST Fernandes, and Andrew Polidore. "An empirical evaluation of undo mechanisms". In: *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*. 2006, pp. 19–27.

[17] *Child accounts*. June 2024. URL: https://www.meta.com/help/quest/articles/accounts/account-settings-and-management/index-child-accounts/.

[18] Isaac Cho, Xiaoyu Wang, and Zachary J Wartell. "HyFinBall: a two-handed, hybrid 2D/3D desktop VR interface for multi-dimensional visualization". In: *Visualization and Data Analysis 2014*. Vol. 9017. SPIE. 2014, pp. 150–163.

[19] *Cobra Logging*. June 2024. URL: https://assetstore.unity.com/packages/tools/utilities/cobra-logging-233277.

[20] Noah Coomer et al. "Evaluating the effects of four VR locomotion methods: joystick, arm-cycling, point-tugging, and teleporting". In: *Proceedings of the 15th ACM symposium on applied perception*. 2018, pp. 1–8.

[21] Christopher Coutinho. "Unity (R) virtual reality development with VRTK4". In: *APress, Berlin, Germany* 1 ().

[22] Carolina Cruz-Neira et al. "The CAVE: audio visual experience automatic virtual environment". In: *Communications of the ACM* 35.6 (1992), pp. 64–73.

[23] Carolina Cruz-Neira et al. "Scientists in wonderland: A report on visualization applications in the CAVE virtual reality environment". In: *Proceedings of 1993 IEEE research properties in virtual reality symposium*. IEEE. 1993, pp. 59–66.

[24] James J Cummings and Jeremy N Bailenson. "How immersive is enough? A meta-analysis of the effect of immersive technology on user presence". In: *Media Psychology* 19.2 (2016), pp. 272–309.

[25] Mallesham Dasari et al. "Scaling VR Video Conferencing". In: *2023 International Conference on Virtual Reality and 3D User Interfaces (VR)*. 2023.

[26] Andreas Dengel et al. "A review on augmented reality authoring toolkits for education". In: *Frontiers in Virtual Reality* 3 (2022), p. 798032.

[27] *Dissonance Unity Voice Chat*. June 2024. URL: `https://placeholder-software.co.uk/dissonance/`.

[28] John Dudley et al. “Performance envelopes of virtual keyboard text input strategies in virtual reality”. In: *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2019, pp. 289–300.

[29] Loren Puchalla Fiore et al. “Towards Enabling More Effective Locomotion in VR Using a Wheelchair-based Motion Platform.” In: *EGVE/EuroVR*. 2013, pp. 83–90.

[30] Anton Franzluebbers and Kyle Johnsen. “Versatile Mixed-method Locomotion under Free-hand and Controller-based Virtual Reality Interfaces”. In: *Proceedings of the 29th ACM Symposium on Virtual Reality Software and Technology*. VRST ’23. Christchurch, New Zealand: Association for Computing Machinery, 2023. ISBN: 9798400703287. DOI: 10.1145/3611659.3615701. URL: https://doi.org/10.1145/3611659.3615701.

[31] Anton Franzluebbers et al. “Collaborative virtual reality training experience for engineering land surveying”. In: *Cross Reality and Data Science in Engineering: Proceedings of the 17th International Conference on Remote Engineering and Virtual Instrumentation 17*. Springer. 2021, pp. 411–426.

[32] Anton Franzluebbers et al. “Virtual Reality Point Cloud Annotation”. In: *Proceedings of the 2022 ACM Symposium on Spatial User Interaction*. SUI ’22. Online, CA, USA: Association for Computing Machinery, 2022. ISBN: 9781450399487. DOI: 10.1145/3565970.3567696. URL: https://doi.org/10.1145/3565970.3567696.

[33] *Free Video Conferencing Software for Web and Mobile*. June 2024. URL: https://jitsi.org/.

[34] Sebastian Friston, Ben Congdon, and Anthony Steed. "Teaching Social Virtual Reality With Ubiq". In: *IEEE Computer Graphics and Applications* 42.6 (2022), pp. 116–122.

[35] Sebastian J Friston et al. "Ubiq: A system to build flexible social virtual reality experiences". In: *Proceedings of the 27th ACM symposium on virtual reality software and technology*. 2021, pp. 1–11.

[36] Ronan Gaugne et al. "Interactive and Immersive Tools for Point Clouds in Archaeology". In: *ICAT-EGVE 2019-International Conference on Artificial Reality and Telexistence-Eurographics Symposium on Virtual Environments*. 2019, pp. 1–8.

[37] Pierre-Yves Gilliéron, Geoffrey Vincent, and Bertrand Merminod. *Blending a MOOCs with interactive teaching*. Tech. rep. 2015.

[38] *Gorilla Tag*. June 2024. URL: `https://www.gorillatagvr.com/`.

[39] *Gorilla Tag crosses 10M VR players and $100M in revenue*. June 2024. URL: `https://venturebeat.com/games/gorilla-tag-crosses-10m-vr-players-and-100m-in-revenue/`.

[40] Simone Grassini and Karin Laumann. "Are modern head-mounted displays sexist? A systematic review on gender differences in HMD-mediated virtual reality". In: *Frontiers in psychology* 11 (2020), p. 1604.

[41] Jens Grubert et al. "Text entry in immersive head-mounted display-based virtual reality using standard keyboards". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2018, pp. 159–166.

[42] Kenny Gruchalla. "Immersive well-path editing: investigating the added value of immersion". In: *IEEE Virtual Reality 2004.* IEEE. 2004, pp. 157–164.

[43] Timo Hackel et al. "Semantic3d. net: A new large-scale point cloud classification benchmark". In: *arXiv preprint arXiv:1704.03847* (2017).

[44] Asim Hameed, Andrew Perkis, and Sebastian Möller. "Evaluating Hand-tracking Interaction for Performing Motor-tasks in VR Learning Environments". In: *2021 13th International Conference on Quality of Multimedia Experience (QoMEX).* 2021, pp. 219–224. DOI: 10.1109/QoMEX51781.2021.9465407.

[45] Warren Harrison. "Eating your own dog food". In: *IEEE Software* 23.3 (2006), pp. 5–7.

[46] Sandra G Hart and Lowell E Staveland. "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research". In: *Advances in psychology.* Vol. 52. Elsevier, 1988, pp. 139–183.

[47] Marlene Huber et al. "Exploring Locomotion Techniques for Seated Virtual Reality". In: *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW).* IEEE. 2023, pp. 925–926.

[48] Robert JK Jacob et al. "Reality-based interaction: a framework for post-WIMP interfaces". In: *Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM. 2008, pp. 201–210.

[49] Muhammad Abid Jamil et al. "Software testing techniques: A literature review". In: *2016 6th international conference on information and communication technology for the Muslim world (ICT4M)*. IEEE. 2016, pp. 177–182.

[50] Patrick Karsh. *Understanding Arguments Against TDD*. Oct. 2023. URL: https://patrickkarsh.medium.com/understanding-arguments-against-tdd-c91de619ce7b.

[51] Rajiv Khadka et al. "Evaluation of collaborative actions to inform design of a remote interactive collaboration framework for immersive data visualizations". In: *International symposium on visual computing*. Springer. 2016, pp. 472–481.

[52] Chaowanan Khundam et al. "A comparative study of interaction time and usability of using controllers and hand tracking in virtual reality training". In: *Informatics*. Vol. 8. 3. MDPI. 2021, p. 60.

[53] Jinwook Kim et al. "Exploration of the Virtual Reality Teleportation Methods Using Hand-Tracking, Eye-Tracking, and EEG". In: *International Journal of Human–Computer Interaction* (Aug. 2022), pp. 1–14. DOI: 10.1080/10447318.2022.2109248.

[54] Pascal Knierim et al. "Physical keyboards in virtual reality: Analysis of typing performance and effects of avatar hands". In: *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018, pp. 1–9.

[55] Boštjan Kovač and Borut Žalik. "Visualization of LIDAR datasets using point-based rendering technique". In: *Computers & Geosciences* 36.11 (2010), pp. 1443–1450.

[56] Oliver Kreylos, Gerald W Bawden, and Louise H Kellogg. "Immersive visualization and analysis of LiDAR data". In: *International Symposium on Visual Computing*. Springer. 2008, pp. 846–855.

[57] Oliver Kreylos and Louise H Kellogg. "Immersive Visual Data Analysis For Geoscience Using Commodity VR Hardware". In: *AGU Fall Meeting Abstracts*. Vol. 2017. 2017, IN32B–04.

[58] Oliver Kreylos et al. "Point-based computing on scanned terrain with LidarViewer". In: *Geosphere* 9.3 (2013), pp. 546–556.

[59] Yoshiki Kudo et al. "Towards balancing VR immersion and bystander awareness". In: *Proceedings of the ACM on Human-Computer Interaction* 5.ISS (2021), pp. 1–22.

[60] Hui-Lung Kuo et al. "FEASIBILITY STUDY: USING A VIRTUAL SURVEYING INSTRUMENT IN SURVEYOR TRAINING". In: *Proceedings of the 2007 International Conference on Engineering Education (ICEE 2007)*. 2007.

[61] Bireswar Laha et al. "Effects of immersion on visual analysis of volume data". In: *IEEE transactions on visualization and computer graphics* 18.4 (2012), pp. 597–606.

[62] Ben Lee et al. "Context-based evaluation of the opus audio codec for spatial audio content in virtual reality". In: *Journal of the Audio Engineering Society* (2023), pp. 145–154.

[63] Christopher Lewis and Frederick C Harris Jr. "Virtual Reality: An Overview, and How to do Typing in VR." In: *International Journal for Computers & Their Applications* 30.1 (2023).

[64] E Li et al. “SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, pp. 1108–1115.

[65] Donghae Lim et al. “Evaluation of User Interfaces for Three-Dimensional Locomotion in Virtual Reality”. In: *Proceedings of the 2022 ACM Symposium on Spatial User Interaction*. 2022, pp. 1–9.

[66] *Logging Package*. June 2024. URL: `https://docs.unity3d.com/Packages/com.unity.logging@1.0/manual/index.html`.

[67] Cho-Chien Lu et al. “Improvement of a computer-based surveyor-training tool using a user-centered approach”. In: *Advanced Engineering Informatics* 23.1 (2009), pp. 81–92.

[68] Cayley MacArthur et al. “You’re making me sick: A systematic review of how virtual reality research considers gender & cybersickness”. In: *Proceedings of the 2021 CHI conference on human factors in computing systems*. 2021, pp. 1–15.

[69] Divine Maloney, Guo Freeman, and Donghee Yvette Wohn. “"" Talking without a Voice" Understanding Non-verbal Communication in Social Virtual Reality”. In: *Proceedings of the ACM on Human-Computer Interaction* 4.CSCW2 (2020), pp. 1–25.

[70] Shady Mansour et al. “BANS: Evaluation of Bystander Awareness Notification Systems for Productivity in VR”. In: *Network and Distributed Systems Security (NDSS) Symposium*. 2023.

[71] Daniel P. Mapes and J. Michael Moshell. "A Two-Handed Interface for Object Manipulation in Virtual Environments". In: *Presence: Teleoperators and Virtual Environments* 4.4 (Nov. 1995), pp. 403–416. DOI: 10.1162/pres.1995.4.4.403. eprint: https://direct.mit.edu/pvar/article-pdf/4/4/403/1622823/pres.1995.4.4.403.pdf. URL: https://doi.org/10.1162/pres.1995.4.4.403.

[72] Oscar Martinez-Rubi et al. "Taming the beast: Free and open-source massive point cloud web visualization". In: *Capturing Reality Forum 2015, 23-25 November 2015, Salzburg, Austria*. The Servey Association. 2015.

[73] Alexander Masurovsky et al. "Controller-Free Hand Tracking for Grab-and-Place Tasks in Immersive Virtual Reality: Design Elements and Their Empirical Study". In: *Multimodal Technologies and Interaction* 4.4 (2020). ISSN: 2414-4088. DOI: 10.3390/mti4040091. URL: https://www.mdpi.com/2414-4088/4/4/91.

[74] Kieran May et al. "3duitk: An opensource toolkit for thirty years of three-dimensional interaction research". In: *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE. 2019, pp. 175–180.

[75] Ryan P McMahan et al. "Evaluating display fidelity and interaction fidelity in a virtual reality game". In: *IEEE transactions on visualization and computer graphics* 18.4 (2012), pp. 626–633.

[76] Michael Melatti and Kyle Johnsen. "Virtual reality mediated instruction and learning". In: *2017 IEEE Virtual Reality Workshop on K-12 Embodied Learning through Virtual & Augmented Reality (KELVAR)*. IEEE. 2017, pp. 1–6.

[77] *Meta Quest Reportedly Had Over 6 Million Monthly Active Users Last October*. June 2024. URL: https://www.roadtovr.com/meta-quest-2-monthly-active-users/.

[78] *Microsoft Azure PlayFab*. June 2024. URL: https://playfab.com/.

[79] Riccardo Monica et al. "Multi-label point cloud annotation by selection of sparse control points". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 301–308.

[80] *MQTT Protocol Specification*. June 2024. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[81] Glenford J Myers, Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[82] Bunta Nakano et al. "Development of a second-screen system for sharing virtual reality information". In: *Software: Practice and Experience* 54.5 (2024), pp. 796–812.

[83] Sophia Neamoniti and Vlasios Kasapakis. "Hand Tracking vs Motion Controllers: The effects on Immersive Virtual Reality Game Experience". In: *2022 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2022, pp. 206–207.

[84] Thinh Nguyen-Vo et al. "NaviBoard and NaviChair: Limited Translation Combined with Full Rotation for Efficient Virtual Locomotion". In: *IEEE Transactions on Visualization and Computer Graphics* 27.1 (2021), pp. 165–177. DOI: 10.1109/TVCG.2019.2935730.

[85] Nels Numan et al. “Ubiq-genie: Leveraging external frameworks for enhanced social vr experiences”. In: *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE. 2023, pp. 497–501.

[86] Joseph O’Hagan and Julie R. Williamson. “Reality aware VR headsets”. In: *Proceedings of the 9TH ACM International Symposium on Pervasive Displays*. PerDis ’20. Manchester, United Kingdom: Association for Computing Machinery, 2020, pp. 9–17. ISBN: 9781450379861. DOI: 10.1145/3393712.3395334. URL: https://doi.org/10.1145/3393712.3395334.

[87] Joseph O’Hagan et al. “Exploring attitudes towards increasing user awareness of reality from within virtual reality”. In: *Proceedings of the 2022 ACM International Conference on Interactive Media Experiences*. 2022, pp. 151–160.

[88] Yun Suen Pai and Kai Kunze. “Armswing: Using arm swings for accessible and immersive navigation in ar/vr spaces”. In: *Proceedings of the 16th international conference on mobile and ubiquitous multimedia*. 2017, pp. 189–198.

[89] Yancheng Pan et al. “Semanticposs: A point cloud dataset with large quantity of dynamic instances”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, pp. 687–693.

[90] Abhishek Patil et al. “The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9552–9557.

[91] Tabitha C Peck, Laura E Sockol, and Sarah M Hancock. "Mind the gap: The underrepresentation of female participants and authors in virtual reality research". In: *IEEE transactions on visualization and computer graphics* 26.5 (2020), pp. 1945–1954.

[92] Nuno Pereira et al. "Arena: The augmented reality edge networking architecture". In: *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2021, pp. 479–488.

[93] Daniela Petrelli, Luigina Ciolfi, and Gabriela Avram. "Envisioning, designing, and rapid prototyping heritage installations with a tangible interaction toolkit". In: *Human–Computer Interaction* 38.2 (2023), pp. 118–158.

[94] *Photon Unity Networking for Unity Multiplayer Games*. June 2024. URL: https://www.photonengine.com/pun.

[95] Christopher Plachetka, Jens Rieken, and Markus Maurer. "The TUBS Road User Dataset: A New LiDAR Dataset and its Application to CNN-based Road User Classification for Automated Vehicles". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2623–2630.

[96] Jussi Rantala et al. "Comparison of Controller-Based Locomotion Techniques for Visual Observation in Virtual Reality". In: *Multimodal Technologies and Interaction* 5.7 (2021). ISSN: 2414-4088. DOI: 10.3390/mti5070031. URL: https://www.mdpi.com/2414-4088/5/7/31.

[97] Julian Rasch et al. "Just Undo It: Exploring Undo Mechanics in Multi-User Virtual Reality". In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–14.

[98] *Rec Room*. June 2024. URL: https://recroom.com/.

[99] Alexander Schafer, Gerd Reis, and Didier Stricker. "Comparing Controller with the Hand Gestures Pinch and Grab for Picking Up and Placing Virtual Objects". In: *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, Mar. 2022. DOI: 10.1109/vrw55335.2022.00220.

[100] Alexander Schäfer, Gerd Reis, and Didier Stricker. "Controlling Teleportation-Based Locomotion in Virtual Reality with Hand Gestures: A Comparative Evaluation of Two-Handed and One-Handed Techniques". In: *Electronics* 10.6 (Mar. 2021), p. 715. DOI: 10.3390/electronics10060715.

[101] Markus Schutz, Katharina Krosl, and Michael Wimmer. "Real-Time Continuous Level of Detail Rendering of Point Clouds". In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, Mar. 2019. DOI: 10.1109/vr.2019.8798284.

[102] Markus Schütz. "Potree: Rendering large point clouds in web browsers". PhD thesis. Wien, 2015.

[103] Markus Schütz et al. "Progressive real-time rendering of one billion points without hierarchical acceleration structures". In: *Computer Graphics Forum*. Vol. 39. 2. Wiley Online Library. 2020, pp. 51–64.

[104] *See your Mac screen on Apple Vision Pro*. June 2024. URL: https://support.apple.com/guide/apple-vision-pro/see-your-mac-screen-tan357ede966/visionos.

[105] *Sentry*. June 2024. URL: https://sentry.io.

[106] Craig R Sherman. "Motion sickness: review of causes and preventive strategies." In: *Journal of travel medicine* 9.5 (2002), pp. 251–256.

[107] William R Sherman et al. "47 Immersive Visualization for the Geological Sciences". In: (2014).

[108] Ruei-Shiue Shiu et al. "Modeling systematic errors for the angle measurement in a virtual surveying instrument". In: *Journal of Surveying Engineering* 137.3 (2011), pp. 81–90.

[109] Anthony Steed et al. "Ubiq-exp: A toolkit to build and run remote and distributed mixed reality experiments". In: *Frontiers in Virtual Reality* 3 (2022), p. 912078.

[110] Darryl George Stuart. "The development of a teaching tool using Sketchup to enhance surveying competence at the Durban University of Technology". PhD thesis. 2015.

[111] Ross Tredinnick, Markus Broecker, and Kevin Ponto. "Progressive feedback point cloud rendering for virtual reality display". In: *2016 IEEE Virtual Reality (VR)*. IEEE. 2016, pp. 301–302.

[112] Alexander James Tuttle, Siddharth Savadatti, and Kyle Johnsen. "Facilitating Collaborative Engineering Analysis Problem Solving in Immersive Virtual Reality". In: *2019 ASEE Annual Conference & Exposition*. https://peer.asee.org/32830. Tampa, Florida: ASEE Conferences, June 2019.

[113] *Use your computer in VR in Meta Horizon Workrooms*. June 2024. URL: https://www.meta.com/help/quest/articles/horizon/getting-started-in-horizon-workrooms/use-computer-in-VR-workrooms/.

[114] Martin Usoh et al. "Using presence questionnaires in reality". In: *Presence: Teleoperators & Virtual Environments* 9.5 (2000), pp. 497–503.

[115] Jean-Marc Valin et al. "High-quality, low-delay music coding in the opus codec". In: *arXiv preprint arXiv:1602.04845* (2016).

[116] Manuel Veit and Antonio Capobianco. "Go'Then'Tag: A 3-D point cloud annotation technique". In: *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE. 2014, pp. 193–194.

[117] *VEL-Connect*. June 2024. URL: https://github.com/velaboratory/VEL-Connect.

[118] *VEL-Share Unity*. June 2024. URL: https://github.com/velaboratory/VEL-Share-Unity.

[119] *VelNet Dissonance Integration*. June 2024. URL: https://github.com/velaboratory/VelNetDissonanceIntegration.

[120] *VelNet Server*. June 2024. URL: https://github.com/velaboratory/VelNetServer.

[121] *VelNet Unity*. June 2024. URL: https://github.com/velaboratory/VelNetUnity.

[122] *VelUtils*. June 2024. URL: https://github.com/velaboratory/VelUtils.

[123] *VelUtils Documentation*. Feb. 2024. URL: https://docs.velutils.ugavel.com/.

[124] *Versioning of Assets and Packages*. June 2024. URL: https://dev.epicgames.com/documentation/unreal-engine/versioning-of-assets-and-packages-in-unreal-engine.

[125] *Virtual Social Platform 'Rec Room' Hits 3 Million Monthly Active VR Users*. June 2024. URL: https://www.roadtovr.com/rec-room-monthly-active-vr-users-3-million-peak/.

[126] *VRChat*. June 2024. URL: https://hello.vrchat.com/.

[127] *VRTK - Virtual Reality Toolkit*. URL: https://www.vrtk.io/.

[128] Greg Wadley and Martin R Gibbs. "Speaking in character: Voice communication in virtual worlds". In: *Online worlds: Convergence of the real and the virtual* (2010), pp. 187–200.

[129] Colin Ware, Kevin Arthur, and Kellogg S Booth. "Fish tank virtual reality". In: *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. 1993, pp. 37–42.

[130] Betsy Williams et al. "Updating orientation in large virtual environments using scaled translational gain". In: *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization*. 2006, pp. 21–28.

[131] Preston Tunnell Wilson et al. "VR locomotion: walking> walking in place> arm swinging". In: *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry-Volume 1*. 2016, pp. 243–249.

[132] Florian Wirth et al. “Pointatme: efficient 3d point cloud labeling in virtual reality”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1693–1698.

[133] Xianshi Xie et al. “A system for exploring large virtual environments that combines scaled translational gain and interventions”. In: *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*. 2010, pp. 65–72.

[134] *XR Interaction Toolkit*. June 2024. URL: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html.

[135] Jimmy F. Zhang et al. “BioVR: a platform for virtual reality assisted biological data integration and visualization”. In: *BMC Bioinformatics* 20.1 (Feb. 2019). DOI: 10.1186/s12859-019-2666-z.

[136] Ruimin Zhang, Bochao Li, and Scott A Kuhl. “Human sensitivity to dynamic translational gains in head-mounted displays”. In: *Proceedings of the 2nd ACM symposium on Spatial user interaction*. 2014, pp. 62–65.

[137] Jingbo Zhao et al. “Comparing hand gestures and a gamepad interface for locomotion in virtual environments”. In: *International Journal of Human-Computer Studies* 166 (Oct. 2022), p. 102868. DOI: 10.1016/j.ijhcs.2022.102868.

[138] Daniel Zielasko et al. “Evaluation of hands-free HMD-based navigation techniques for immersive data analysis”. In: *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE. 2016, pp. 113–119.

[139] SM Zolanvari et al. “DublinCity: Annotated LiDAR point cloud and its applications”. In: *arXiv preprint arXiv:1909.03613* (2019).