University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Hand-tracked 3D Data Selection of Point Clouds in XR

Master's Thesis

Master of Science in Engineering with
Specialisation in Computer Science

Institute of Interactive Technologies

Author
Luca Fluri
luca.fluri@fhnw.ch

Advisor
Prof. Dr. Arzu Çöltekin
arzu.coltekin@fhnw.ch

Expert
Dr. Stefan Arisona
robot@arisona.ch

Brugg-Windisch
August 23, 2024

MSE | MASTER OF SCIENCE IN ENGINEERING

Abstract

Three-dimensional data is crucial in fields like scientific visualization, medical imaging, and astronomy, yet effective 3D selection remains a challenging and unsolved problem. Traditional methods, primarily limited by 2D interfaces, often lack the precision and flexibility required for accurate 3D selection. Additionally, existing 3D techniques are limited to simple selection volumes and must be optimized for point clouds, further complicating the process.

This project addresses these challenges by developing a GPU-based selection approach using Signed Distance Fields (SDFs). This approach enables efficient, real-time selection of near-arbitrary subsets of point clouds in XR environments. The solution is implemented in Unity and integrates novel and existing selection techniques, validated through a user experiment. Results revealed significant performance differences between selection techniques, task complexities, and point clouds. Participants favored brushing modes over direct selection methods — a preference confirmed by quantitative data. This research bridges the gap between traditional 2D methods and XR, offering a robust and efficient approach to 3D data selection.

Keywords:
Point Clouds • Data Selection • XR • Hand Tracking • Signed Distance Fields • User Experience

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Dr. Arzu Çöltekin, for her invaluable guidance, support, and encouragement throughout this project. Her expertise and insights have been instrumental in shaping this work, and I am truly grateful for the opportunity to learn from her.

I would also like to thank all the proofreaders who took the time to review my work and provide constructive feedback. Your careful attention to detail and thoughtful suggestions have greatly improved the quality of this thesis.

A special thanks goes to my family for their unwavering patience, understanding, and support during this journey. Your belief in me has been a constant source of motivation, and I could not have completed this work without your love and encouragement.

Table of Contents

1 Introduction

Three-dimensional data is becoming increasingly important in various fields, such as scientific visualization, medical imaging, or astronomy [1, 2]. The ability to interact with and select specific parts of volumetric data is crucial for exploratory data analysis, research, and annotation tasks. In parallel, recent advancements in XR (Extended Reality) have enabled more intuitive and immersive ways to interact with data. Specifically, hand-tracking technology has seen massive leaps in recent years, allowing for more natural interactions with virtual objects.

Current state-of-the-art solutions for 3D data selection are often limited to classical 2D applications [3–5] that require the user to view 3D data in a 2D space. While this is possible, and 2D solutions are powerful, they often rely on complex interfaces and are computationally expensive, making real-time applications impractical. Additionally, they lack the intuitive and immersive experience that XR can provide. Changing the user's viewpoint in 2D space can be cumbersome and time-consuming. It would be much more natural and direct in a 3D XR environment. The added depth information and, therefore, less occlusion of data in 3D additionally make the data exploration and selection process more intuitive, faster, and more accurate [6, 7]. Products that offer 3D selection of point clouds do not exist in XR, but notable research work has been done in this area (see chapter 2). At this time, 3D selection of point clouds is still a challenging and unsolved problem.

This research is necessary due to three main reasons:

(a) 3D selection is relevant to many scientific and practical use cases.
(b) 3D selection is a complex problem that is still unsolved.
(c) New developments in XR offer promising leads.

Bridging the gap between well-established 2D methods and new XR technologies, this project aims to create a intuitive, effective, and immersive way to interact with point clouds in 3D space. This project has the potential to significantly enhance the user experience and improve the effectiveness of data analysis and exploration.

This project focuses on 3D point cloud selection and addresses these challenges by developing a GPU-based (Graphical Processing Unit) selection approach using Signed Distance Fields (SDFs) to approximate selection volumes. This solution has two main components: First, implementing a GPU-based SDF approach and validating several selection techniques based on that underlying framework. The SDF approach uses the computational power of GPUs to efficiently compute and render SDFs, which serve as selection volumes. These volumes allow users to intuitively select regions of interest within the 3D data by defining arbitrary spatial boundaries around desired points, providing high precision, and handling complex shapes and sizes. The GPU is also used for the selection calculation, leveraging its parallel nature for efficiency.

The second component involves implementing and validating user-friendly selection techniques. The project aims to enhance the user experience in XR environments by combining the GPU-based SDF approach with these techniques.

This project builds upon previous work [8], integrating a GPU-based SDF approach and validating multiple selection techniques. The goal is to develop a system that performs computationally efficiently and provides an intuitive and effective user experience for interacting with 3D data in XR environments.

Based upon the aforementioned problem statement and this project's approach, the following research questions are addressed and answered in this project:

RQ1 How can points inside an arbitrary selection volume be computationally selected and queried in a fast manner?

RQ2 What are intuitive and effective manual (hand-tracked) selection techniques for the user?

The main contribution of this thesis lies in developing a comprehensive prototype, which includes its conceptualization, design, implementation, and testing. This prototype, utilizing SDFs in a GPU accelerated selection approach, effectively demonstrates an efficient and precise method for real-time 3D data selection of point clouds. Furthermore, the thesis extends its contributions by collecting and analyzing user experience data from a cohort of 28 participants, offering insights into the effectiveness of various selection techniques. Two of the four developed selection techniques are novel, and the other two are based on existing research. This synthesis of technical innovation and empirical data collection enhances the understanding and practice of 3D data interaction. It establishes a foundation for future research and applications in this rapidly evolving field.

In the following chapter, I will present related work in XR, point cloud rendering, SDFs, 3D selection techniques and user studies. Then, I will describe the prototype implementation, including the GPU-based SDF approach and the selection techniques. I will discuss the prototype's evaluation in detail, focusing on performance and usability. Finally, I will conclude with a discussion of the results and outline future work in this area.

2 Related Work

2.1 Extended Reality

Extended Reality (XR) is an umbrella term that encompasses Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). These technologies blend the real and virtual worlds, offering immersive experiences that can either fully replace the user's environment (as in VR) or enhance it with digital elements (as in AR and MR) [9]. The rapid advancement of XR technologies has seen their application across a wide range of domains [10–13]. XR's ability to provide interactive and immersive experiences makes it a powerful tool for visualizing complex data and creating engaging user interfaces [14, 15].

XR headsets are one of the primary hardware through which users experience immersive environments, but smartphones are also common for AR applications. These headsets come in various forms, including tethered, standalone, and AR glasses, each with distinct capabilities and use cases.

- **Tethered Headsets:** Devices like the HTC Vive and Oculus Rift fall into this category [16]. They require a connection to a powerful computer to deliver high-quality graphics and processing power. These headsets typically offer higher-fidelity experiences with more robust tracking capabilities.
- **Standalone Headsets:** Standalone devices, such as the Pico, Meta Quest, and Apple Vision Pro series, are self-contained and do not require a connection to a PC. These headsets balance portability and performance, making them accessible for consumer and professional use. The integration of inside-out tracking (where cameras on the headset track the environment and the user's position) and advanced hand-tracking features have made standalone headsets increasingly popular in gaming and enterprise settings. Most Android-based standalone headsets can also be used in a tethered mode.
- **Augmented Reality Glasses & Smartphones:** Standalone AR devices, like the Microsoft HoloLens, Magic Leap, or AR smartphone apps, overlay digital content onto the real world, allowing users to interact with virtual objects while remaining aware of their physical surroundings. These headsets are particularly useful in fields like architecture, where virtual models can be viewed in real-world environments, or in remote assistance, where experts can guide users through tasks with visual overlays.

Significant advancements in display technology, sensor integration, and processing power have marked the evolution of XR headsets. Modern headsets are equipped with high-resolution displays, low-latency tracking systems, and increasingly sophisticated sensors that enhance the immersion and interactivity of XR experiences. For instance, advancements in foveated rendering (which reduces the rendering workload by focusing high resolution only where the eye is looking) [17, 18] have contributed to more realistic and performant experiences.

Hand-tracking is a critical component of XR that allows users to interact with virtual objects using their natural hand movements without needing traditional controllers. This technology leverages a combination of computer vision and machine learning algorithms to detect and interpret the position, orientation, and movement of the user's hands and fingers in real-time.

Early implementations of hand-tracking in XR relied on external hardware, such as Leap Motion devices, which used infrared sensors to track hand movements. However, as XR headsets have evolved, so too has the integration of hand-tracking technology. Modern headsets like the Meta Quest and Microsoft HoloLens incorporate built-in cameras and sensors, enabling more sophisticated and accurate hand-tracking capabilities without additional peripherals.

The role of hand-tracking in XR is particularly significant in fields that require precise manipulation of virtual objects, such as medical simulations, where surgeons can practice procedures using their hands, or in design and engineering, where 3D models can be manipulated directly. In these contexts, the naturalness of hand-tracking reduces the learning curve and increases user engagement, making it a critical area of research and development in XR [19, 20].

XR development is a rapidly evolved landscape, driven by the advancements of hardware and softare. Key players in this field include game engines like Unity [21] and Unreal Engine [22], which are industry standards for creating XR experiences. Unity is widely favored and has higher market share in the XR space [23]. Unity excels at ease of use, flexiblity and robust support of XR platforms. Unreal Engine on the other hand, is known for its high-fidelity graphics and advanced rendering capabilities.

2.2 Point Cloud Rendering

Efficient data structures are fundamental to managing large point clouds, enabling the handling of millions or even billions of points with high performance. Structures such as kd-trees [24], quadtrees [25], Octrees [26–28], Bounding Volume Hierarchies (BVH) [29], and tools like OpenVDB [30] are commonly used to organize point data spatially, allowing for rapid search and retrieval operations. These data structures optimize both rendering and selection processes by significantly reducing the complexity of point queries and enabling efficient data traversal [24–29].

Beyond efficient data storage, rendering techniques play an equally critical role in achieving high-performance visualization of point clouds. Techniques such as frustum culling, level-of-detail (LOD) rendering, and point cloud compression [27, 28, 31, 32] are pivotal in enhancing rendering efficiency and reducing computational overhead. Frustum culling minimizes the number of points rendered by eliminating those outside the viewer's field of view, while LOD rendering dynamically adjusts the detail level of the point cloud based on the viewer's distance, balancing performance with visual quality. Point cloud compression further reduces the data size, making it more feasible to render large-scale datasets in real-time.

Recent advancements in point cloud rendering have introduced out-of-core[33] techniques combined with LOD rendering, supporting the visualization of massive point clouds that far exceed the memory capacity of a typical GPU. Out-of-core methods store the majority of the point cloud data on disk, loading only the necessary subsets into memory as needed, which is crucial for handling large-scale datasets without compromising performance. This approach, often implemented alongside octree-based data structures, has proven particularly effective in web-based applications [26, 34], where it allows for the visualization of extensive point clouds without requiring substantial local computational resources. These methods have gained popularity in various domains, including geospatial mapping, urban planning, and cultural heritage preservation [26], where they enable the detailed visualization of complex environments.

Unity-based approaches to point cloud rendering also exist [28, 27], though they are often not publicly available or fully production-ready. While Unity provides a flexible environment for developing interactive applications, its out-of-the-box capabilities for handling large point clouds are limited compared to more specialized engines or frameworks. Some efforts have been made to integrate efficient point cloud rendering techniques within Unity, leveraging its robust ecosystem and compatibility with XR devices, but these solutions typically require significant customization and are not yet as mature as those found in other platforms. As such, while Unity holds promise for future developments in point cloud rendering, particularly in XR applications, it remains a developing area of research and development.

2.3 Signed Distance Fields

Signed distance fields are essential in computer graphics and computational geometry for representing shapes and calculating distances. They store, for each texture pixel (texel), the orthogonal distance to the surface of an object, along with a sign indicating whether the point is inside or outside the object. By convention, this distance is positive outside and negative inside. In figure 1, the SDF of a circle is visualized, with the distance values encoded as colors.

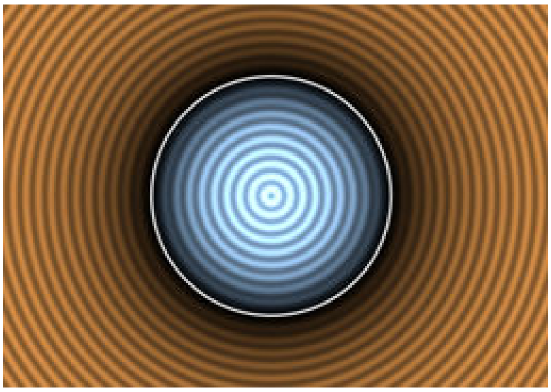


Figure 1: Signed Distance Field of a Circle. Negative values (blue) are inside, positive values (orange) are outside, and black represents the surface. [35]

The foundational work on Signed Distance Fields (SDFs) can be traced back to the development of the Euclidean distance transform for binary images in the 1980s. This transform was a significant breakthrough, enabling the calculation of the shortest distance from each pixel to the nearest object boundary in binary images [36]. These early developments primarily focused on image processing and pattern recognition, laying the groundwork for later 3D graphics and modeling applications.

The advent of real-time ray marching algorithms further propelled the use of SDFs. These algorithms, which leverage the parallel processing capabilities of modern GPUs, allow for the efficient rendering of implicit surfaces. This advancement was crucial for procedural content generation and real-time graphics applications, enabling the visualization of highly detailed and dynamic scenes [37].

Signed distance fields have played a pivotal role in various areas such as three-dimensional rendering, collision detection [38, 39], shape reconstruction, and mesh generation [40, 41]. They offer a compact and efficient representation of geometric shapes, facilitating operations such as anti-aliased rendering and gradient estimation[42]

Researchers have explored the use of signed distance fields in ray tracing, image processing, and solid modeling [43–46]. Their ability to represent complex shapes and smoothly interpolate between surfaces makes them valuable in computer graphics and related fields.

Several techniques have been developed to efficiently construct and manipulate signed distance fields, including distance estimation from triangular meshes and various optimization approaches [47–51]. Tools for the Unity game engine are also available [52–54].

2.4 3D Selection Techniques

Manual techniques for selecting subsets of data within a point cloud are essential for data exploration, annotation, and analysis tasks. Traditional techniques often involve selecting points within a 2D projection, which can be limiting and less intuitive [55, 56]. More advanced methods include 3D selection volumes and tools that allow users to specify regions of interest directly within the volumetric space [57, 58, 28].

Selection techniques in XR environments have evolved to leverage hand-tracking capabilities and other input modalities. Methods such as ray-casting, where users point and select objects using a virtual ray emitted from their hands [59], and volume-based selection [57, 58, 28], where users define a spatial boundary around the desired points, have become popular. These techniques provide precise and efficient ways to interact with 3D data, enhancing usability and user satisfaction [60, 19].

Early work in 3D data selection developed a bi-manual point cloud selection tool using a stereo camera for hand-tracking and a head-mounted display, pioneering cost-effective VR-based 3D data visualization [61]. This system allowed users to translate, scale, and rotate point clouds, addressing issues like the “fat finger problem” [62]. Despite its innovation, it was limited to single-point selection and had minimal user testing.

Later improvements integrated Leap Motion with a Unity application for 3D input while users viewed data on a 2D screen [56]. This system included modes for transformations and selections controlled via hand gestures. However, the Leap Motion’s tracking limitations and the lack of depth perception on a 2D screen posed challenges.

Another technique, Slice-n-Swipe, used a “chef’s knife” metaphor with Leap Motion, allowing users to cut and select subsets of point clouds [63]. Though innovative, it required a 3D mouse for spatial movement, highlighting the limitations of hand-tracking technology at the time.

The Tangible Brush combined 2D touch input with 6 degrees of freedom (DOF) 3D tangible input, using a spatially aware tablet to perform 3D selections [64–66]. While more accurate than existing methods, it was limited by 2D output and lacked the advantages of viewing 3D data in a 3D space. This technique suggested further research integrating modern XR technology.

Recent volumetric selection techniques are based upon simple geometric shapes like spheres, cuboids, or cylinders [57, 58, 28]. Various use cases have validated these techniques, from trajectory selection[57] to point-cloud brushing with spherical volumes [58, 28]. Built with Unity, these techniques have limited evaluation with more extensive and denser point clouds and have been evaluated with trajectory data sets[57] or with a discrete low number of game objects to approximate points in Unity [58]. However, significant work has been done to extend the capabilities of Unity in point cloud rendering [28], which has also allowed for spherical collision-based selection with Unity's engine physics with a massive number of points. This collision approach relied on the query possibilities of the underlying data structure and only allowed for spherical selection volumes.

Automatic techniques are an efficient alternative to manual selection methods, particularly when dealing with massive data sets and point clouds [67]. The techniques leverage machine learning and advanced segmentation methods to automate the selection of relevant points, reducing time and effort from users [67–70].

One major issue for machine learning approaches is their inherent dependency on large, high-quality datasets [71, 67]. The final selection accuracy is heavily influenced by the diversity and representativeness of training data, potentially degrading selection results. Another challenge is the computational complexity of algorithmic techniques [68, 69], potentially requiring extensive pre-calculation and data preparation.

To summarize, combining point cloud rendering, efficient data structures, and advanced selection techniques with XR and hand-tracking technologies shows promise for enhancing 3D data interaction. While automatic approaches do exist, they are not ready for real-time XR use, and there is still a significant need for accurate and flexible manual approaches. This project aims to improve the performance and usability of 3D data selection in XR environments by integrating GPU-based SDFs for selection volumes and validating user-friendly manual selection techniques.

2.5 User Studies

User studies are crucial in understanding how individuals interact with Extended Reality (XR) technologies. These studies contribute to designing and developing user-centered XR systems by measuring usability, presence, and task performance. They typically employ qualitative and quantitative methods, adapted from traditional Human-Computer Interaction (HCI) research but tailored to the requirements of immersive environments [72, 73].

A common approach in XR user studies is an experimental design, where researchers systematically vary elements of the XR system—such as interaction techniques or levels of immersion, to observe their effects on user experience. These experiments often utilize within-subjects designs to minimize variability and isolate the impact of the tested variables [74]. To complement this, surveys and questionnaires, like the Presence Questionnaire (PQ), Immersive Tendencies Questionnaire (ITQ), or the System Usability Scale (SUS), are widely used to capture subjective user feedback, providing insights into users subjective experience [75].

Task performance metrics, such as accuracy, completion time, and error rates, are critical in XR studies focused on professional applications or training scenarios. These metrics help evaluate how effectively users can perform tasks in an XR environment, essential for validating the system's utility in real-world contexts.

Qualitative methods, including interviews and focus groups, are also integral to XR user studies. These methods allow researchers to delve deeper into users' thoughts and feelings, uncovering insights that might not emerge through quantitative measures alone. For example, interviews conducted before and after XR experiences can reveal changes in user perceptions and attitudes, offering valuable information for refining system design.

However, conducting user studies in XR presents several challenges. Motion sickness, a common issue in immersive environments, can affect the validity of the results and needs to be carefully managed [76]. The complex datasets generated by XR studies, mainly from behavioral and physiological measures, require sophisticated analysis techniques to derive meaningful insights [77].

Finally, the insights gained from XR user studies are invaluable. They not only guide the development of more intuitive and accessible XR systems but also contribute to a deeper understanding of how these technologies impact users. The findings from XR user studies are essential for ensuring that these immersive technologies fulfill their potential to enhance user experiences and achieve their intended outcomes [78].

3 Prototype

This chapter first provides an overview of the development methods and technological concept and then details the Unity implementation with all its related components.

3.1 Methods and Concept

Development was based on iterative user feedback in a qualitative manner and an agile software development process. Based upon an initial research phase, the prototype was conceptualized and developed. The prototype's computational performance was tested locally, and the results are presented in chapter 5. The prototype was tested with a cohort of 28 participants in a user experiment to gather qualitative and quantitative feedback on the effectiveness and usability of the developed selection techniques. The results of this study are presented in chapter 5.

The state-of-the-art of point cloud selection has the following challenges (see related work in chapter 2):

(a) Traditionally, 2D software tools have no stereo 3D viewer and little to no recent adoption of XR technology.
(b) Automatic selection approaches do exist (often based on segmentation models) but are inherently data-dependent and do not always work perfectly, necessitating manual selection.
(c) Manual selection approaches often rely on simple, less precise, and adaptable techniques like bounding boxes or ray-casting, limiting possible selection volumes and requiring extensive optimization for semi-large point clouds.
(d) Selection in large point clouds is traditionally computationally intensive and not optimized for real-time application, let alone frame-by-frame calculation.

Regarding (a) and (b), a Unity-based XR application that leverages the advantages of viewing 3D data in 3D and enables natural, manual interaction through hand-tracking seems more appropriate. Addressing (c) and (d), a SDF-based selection algorithm that fully utilizes the GPU's parallelization power and enables near-arbitrary selection volumes is proposed. The concept above, combined with a fast and efficient point cloud render approach and several volumetric selection techniques, enables accurate real-time selection and querying of points inside near-arbitrary volumes.

Unity provides a robust framework for developing XR applications as the underlying game engine and is next to the Unreal Engine [22] and Godot [79], the industry standard for XR development. Using Unity's XR Interaction Toolkit (XRI) and OpenXR as the XR plugin provider, it is possible to create a platform-independent XR application that can run on various devices. Although the prototype was developed and tested on a Meta Quest Pro and Quest 3 headset in practice, the use of OpenXR ensures compatibility with other hand-tracking-enabled devices. While this work is not Unity-specific and could be implemented in other engines, Unity also provides a robust ecosystem and little friction to get started with XR development.

SDFs can represent an approximation of any 3D mesh in a compact numerical format with less memory than a complex polygonal mesh. Besides memory advantages, SDFs can be stored as 3D textures and are natively supported on the GPU, making SDFs an excellent choice for a volumetric selection. State-of-the-art approaches to point cloud selection rely on complex data-structures and simple selection shapes like spheres or boxes that facilitate fast data traversal. With this SDF based approach on the other hand, arbitrary selection volumes are possible.

This proposed selection concept makes full use of GPU-based compute shaders. Using only a compute shader that runs over all points of the point cloud, checking whether this point is inside or outside a given SDF texture, and repeating that per selection volume, it is possible to fully parallelize and drastically simplify this simple selection task by combining it with the render pipeline with little additional cost. The result of each step of the selection compute shader is then saved into a compute buffer, which allows access outside the GPU. Limiting the data transfer between CPU and GPU is paramount to avoid bottlenecking the calculations with data transfers.

Visualizing large point clouds is a computational challenge on its own. While state-of-the-art approaches and optimizations do already exist [26–28], during experimentation and pilot tests, it was found that using an indirect and instanced rendering approach in Unity, using direct GPU draw calls [27], it is possible to draw millions of points and display them on an XR headset using multi-pass rendering, which still effectively cuts the frame rate in half. This is fast enough to evaluate the SDF-based selection approach and build an efficient prototype. Recent state-of-the-art approaches have been considered but deemed unnecessary due to their complexity and stability in Unity adaptations. This project's focus is foremost selection, therefore implementing a performant massive point cloud renderer in Unity was out-of-scope, but is essential future work.

Hand-tracking was chosen as the sole input modality for this research due to its potential to offer a more natural and intuitive interaction experience in XR environments. Unlike traditional controllers, hand-tracking allows users to interact with virtual objects directly with their hands, mimicking real-world gestures and actions, which can significantly reduce the learning curve and increase user engagement. Research has shown that hand-tracking can enhance spatial awareness and immersion, making it a powerful tool for tasks that involve complex spatial manipulations, such as 3D data selection [19, 80, 81, 60]. Additionally, recent advancements in hand-tracking technology, such as those by Ultraleap, have improved the accuracy and responsiveness of these systems, addressing previous limitations related to tracking fidelity and occlusion [82].

In order to validate and test the SDF-based approach, several volumetric selection techniques have been developed and integrated into the prototype. In the first (*Shapes*), users can spawn several shapes, change their dimensions, and place them inside the point cloud as selectors (based upon previous and related work [83, 84, 56, 57, 85]). In the second (*Convex Hull*), a convex hull can be drawn while pinching and drawing in the air (a novel technique). The third and fourth modes are brush modes, whereas, in the third (*Brush Sphere*), users can brush/draw with a sphere (based upon related work [58, 86, 28]). Furthermore, in the fourth (*Brush Hands*), it is possible to directly use the visualized hand mesh as a selection shape (a novel technique). In addition to the selection modes, a hand-attached UI is available to the user to change point size, change selection modes, and reset or accept the current selection. Finally, users can transform the point cloud and all active selection shapes using a two-handed fist gesture. This allows for intuitive and natural interaction with the point cloud and the selection shapes.

3.2 Implementation

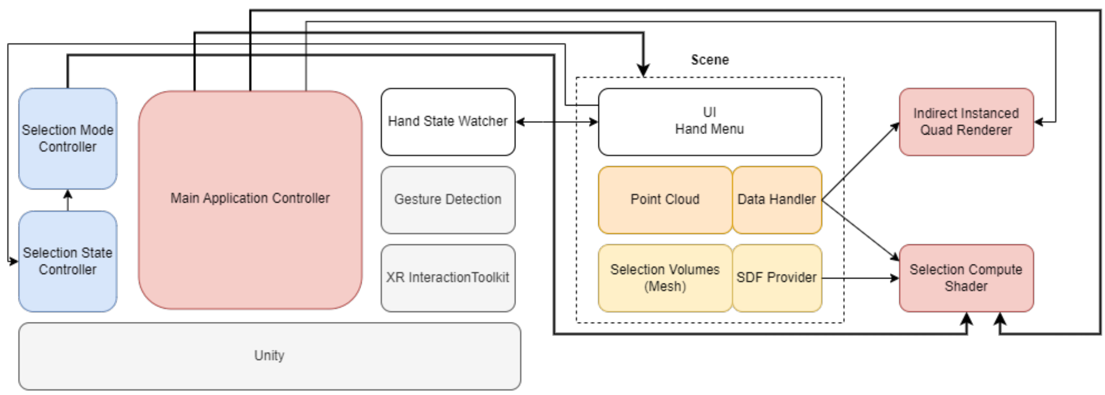


Figure 2: System Architecture of the Prototype

The application is built with Unity using OpenXR and Unity's XR Interaction Toolkit. A central controller script in the scene handles everything from point cloud and selector mesh handling to selection compute shader dispatch and GPU draw calls. On top of that, several empty game objects represent each selection technique/mode. Each selection mode has all its necessary interaction scripts attached to it. A selection state controller controls all four of these selection mode controllers. This state controller effectively controls the active state of each mode controller. This is possible as scripts are only activated if their parent game object is active.

In the scene, only selector shapes and the point cloud are visible, apart from the tracked hand models and the hand UI. Regarding the hand-tracking, there is a seperate "Hand Watcher" script that keeps track of position and gesture state for both hands and relevant finger joints based upon XR Interaction Toolkit's gesture detection and hand data. All selection techniques access and reference this script when checking for the state of hands and gestures.

The application consists of three main components:
- Selection Handling
- Point Cloud Rendering
- Hand-tracking-based user interaction

The following sections will explain each of these components and all other relevant scripts and procedures in detail.

3.2.1 Selection Handling

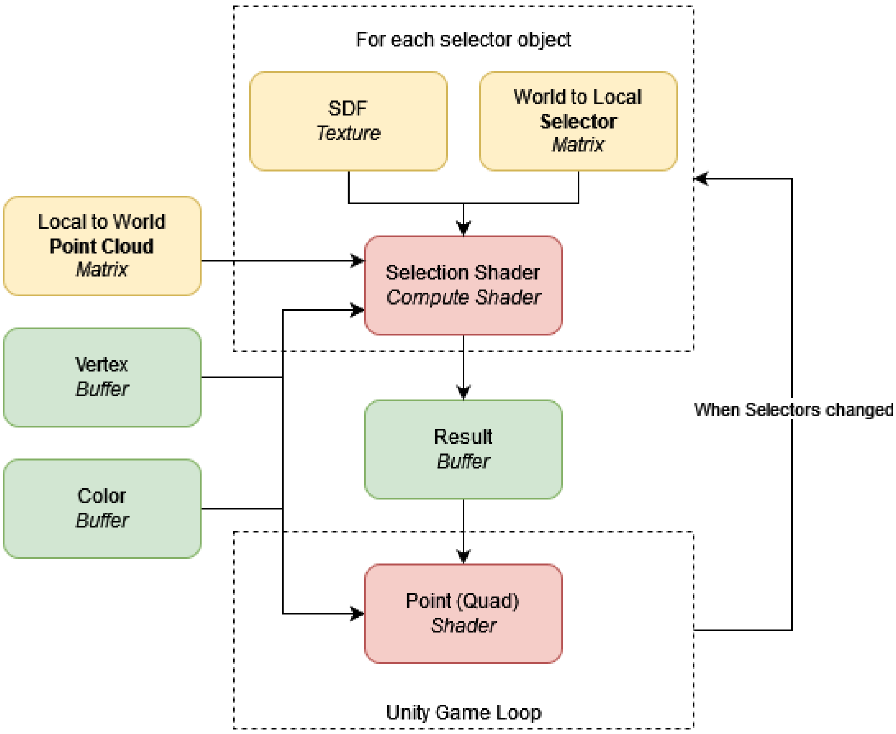


Figure 3: Overview of the selection handling. Yellow is data, red are shaders and green are data buffers

In this concept, selections are volumetrically defined as meshes for the user. Using these selection meshes, an SDF texture is calculated for each selector (covered more in detail in section 3.2.1.1). Assuming an SDF is given for an arbitrary mesh selector, it is trivial to check whether a point is inside or outside that SDF by simply sampling the SDF texture at the correct position.

To achieve this, all local vertex positions and colors of the point cloud are stored in separate buffers on the GPU (Shown in green in figure 3). The selection handling happens in a compute shader call for every selector and every time a selector changes. In this compute shader (Selection shader in red in figure 3), the local point from the vertex buffer is first transformed to Unity's scene world space and then transformed into local SDF space of the selector using both transform matrices as shown in figure 3 in yellow. Then, the SDF texture of the current selector is sampled, and the result is saved in a new results buffer solely responsible for keeping track of the inside/outside state for each point of the point cloud. Given that the result buffer stays in GPU memory, the same buffer is used to color the point cloud accordingly in the render stage using a custom shader (more on that in section 3.2.2), which eliminates unnecessary data transfer. Data from the result buffer is available to transfer out of the GPU memory therefore allowing the application to know exactly which points are selection and which are not.

3.2.1.1 SDF Generation

Signed Distance Fields (SDFs) are powerful tools for defining volumetric regions within 3D spaces. SDFs represent the distance of any point in space to the nearest surface, with the sign indicating whether the point is inside or outside the surface. SDFs are an analytical approximation of the surface, and it is therefore paramount for the SDF to be as accurate as possible in order to be used for selection purposes.

While generating an SDF per se is not difficult, generating an SDF from an arbitrary mesh that is accurate and fast enough for real-time applications is far more challenging. In early development, it was found that meshes with sharp corners tend to have difficulties being generated. Also, in some cases, a sign could flip during calculation, propagating through the texture and rendering the texture practically useless for selection purposes. Each selector volume in the prototype has a component script that is solely responsible for serving an SDF from its local mesh. This way, the source of the final SDF texture is abstracted enough so that different SDF providers could theoretically be used. In practice, though, only the SDF Baker from Unity's VFX Graph is used due to its performance and accuracy compared to multiple other solutions tested [52, 53].

The SDF Baker from Unity's VFX Graph [54] is developed for real-time visual effects (VFX) applications and is, therefore, highly optimized. It takes advantage of compute shaders to calculate the texture, making it an excellent choice for this project. It aligns with the technological concept and allows for real-time SDF generation of arbitrary meshes.

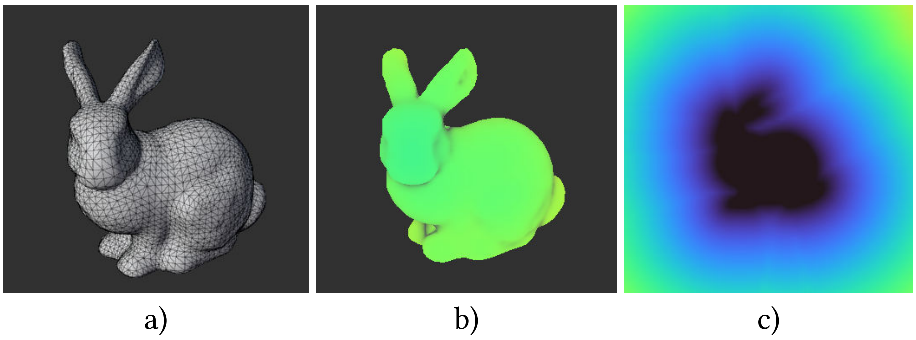


Figure 4: The Stanford bunny represented as mesh in a). A 3D visualization of the generated SDF in b) and a 2D slice of the same SDF in sub figure c)

Each selector has a component solely responsible for serving an up-to-date SDF of its mesh. Figure 4 shows an example mesh with two visualizations for its SDF generated by the SDF Baker of the VFX Graph.

3.2.2 Point Cloud Rendering

Point clouds can easily range between a few thousand and hundreds of millions of points. While rendering a few thousand points is trivial, rendering millions of points in real-time is challenging.

Traditionally, data structures like Octrees and Level of Detail (LOD) optimize the rendering process and render only the points that can be viewed with the appropriate quality. Potree [26] is a popular open-source solution that combines Octrees and LOD to render massive point clouds in the browser. While Potree is an excellent and widely used solution in the remote sensing community, it is unsuitable for real-time applications due to its complexity and rigidity, as it requires point clouds to be converted into a custom format first. There is an attempt at porting a similar approach to Unity [27, 28], but this solution still needs to be production-ready and suffers from many bugs.

Point clouds are loaded into the application using either a mesh based approach where PLY files [87] are converted into a mesh object using a PLY importer[88] or via a CSV[89] approach where the point cloud is read directly from file. In both supported cases a "Point Data Handler" script is used that serves a list of vertices positions and colors. The PLY approach uses a modified PLY importer to allow for readable mesh data [88] and the CSV approach uses a self-built CSV parser for position and color values.

This prototype renders each point as an instanced and rotated quad. This simple and efficient approach allows millions of points to be rendered in real-time. The indirect rendering further optimizes the process by reducing the number of draw calls and allowing the GPU to handle the rendering process more efficiently [27]. The result is a visually appealing representation of the point cloud data that maintains high performance and interactivity.

To fully leverage the power of the GPU, it is paramount to refrain from transferring data between CPU and GPU as much as possible since this data transfer is very slow. To avoid this, all data is stored in the GPU in buffers, and all calculations are done on the GPU using compute shaders (Section 3.2.1).

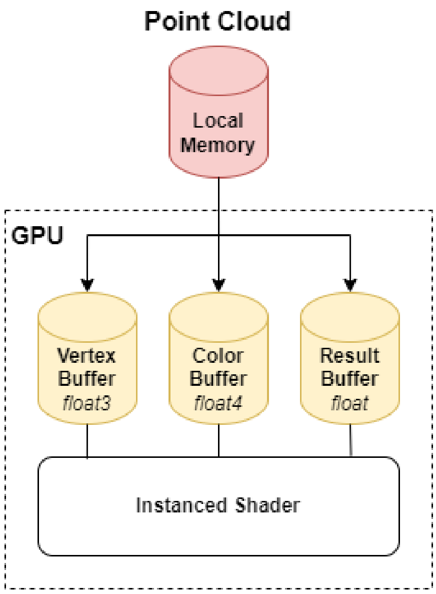


Figure 5: Diagram showing how the point cloud data gets split into three seperate compute buffers on the GPU. The instanced rendering shader then accesses these buffers.

Instead, all points and their respective colors are stored and kept in buffers on the GPU (Figure 5). This allows for minimal data transfer between CPU and GPU after initialization. Points are rendered via direct GPU draw calls using Unity's Graphics API with `Graphics.RenderMeshIndirect` (Instanced Shader in figure 5). This draw call uses a vertex position buffer, color buffer, and result buffer on the GPU, allowing for the rendering of millions of points as quads without any further optimizations.

3.2.3 User Interaction

Given the rise and improvement of hand-tracking technology in recent years, this work leverages hand-tracking as the only input method for the user. OpenXR is the underlying XR framework and enables several different hand-tracking headsets to be used as input.

User interactions in this concept can then be divided into three categories:
- Hand Menu (UI)
- Point Cloud Transformations
- Selection Techniques

3.2.3.1 Hand Menu

In this prototype, a user can open a hand menu by looking at their left hand, and it can then be navigated using a direct poking gesture with the right hand, allowing for a similar "touch" interaction as with traditional touchscreens. Users can change the displayed point size in the menu, reset or accept the current solution, and switch between several selection modes (Fig. 6).

Figure 6: Hand menu of the prototype, activated by looking at open palm.

The XR Interaction Toolkit by Unity, which ties in directly with the OpenXR Runtime, is used as the UI and interaction provider, which greatly simplifies development and future UI extensions.

3.2.3.2 Point Cloud Transformations

It is paramount to allow users to rotate, scale, and translate the point cloud to their liking. This is important for exploratory analysis and the accurate selection of subsets as the required and intended region of interest is not always visible.

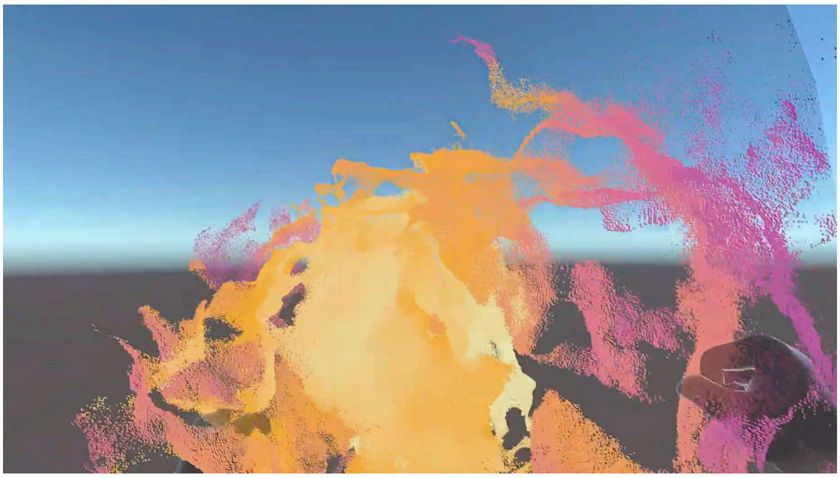


Figure 7: Transforming (translate, scale and rotate) the point cloud using a two-handed fist gesture.

To facilitate this interaction, users can always use a two-handed fist gesture. This then maps the distance between the hands to the scale of the point cloud. For translation, the center point between both hands is mapped to the location, and for rotation, the angle between the hands is mapped to the local rotation of the point cloud. This allows a very intuitive and natural way to manipulate the point cloud in 3D space. A two-handed fist gesture was used, as this is very natural and distinct from pinch gestures used in all other selection techniques, reducing the possibility of false positives.

3.2.3.3 Selection Techniques

Four different selection techniques have been implemented in this prototype to test the feasibility of different selection techniques and the SDF selection itself and to provide the user with a variety of options. The first two techniques are strictly direct, and the last two are continuous brushing techniques.

Figure 8: The “pinch and drag apart” gesture used to spawn shapes.

For the first selection technique (*Shapes*), simple shapes can be spawned by pinching closely together and dragging the hands apart (Fig. 8). This shape based approach is inspired by related work [83, 84, 56, 57, 85]. This gesture spawns a preselected shape (sphere, box, or cylinder) between the hands and directly locks the distance between the hands to the object’s scale. Boxes and cylinders are scaled per axis, while the sphere is scaled uniformly. This allows the users to scale the object and change its initial dimensions as long as both hands are pinching. As soon as the pinch gesture ends, dimensions are locked, and the object can only moved, rotated, and uniformly scaled by naturally grabbing it with either one or two hands.

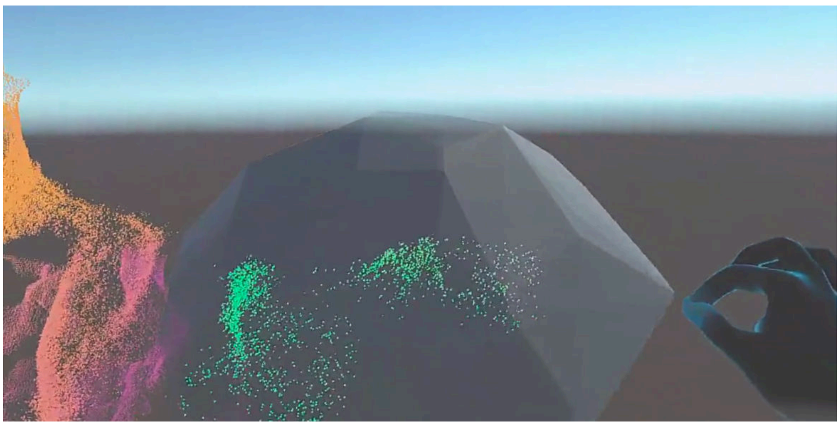


Figure 9: The drawing of convex hulls in action using the pinch gesture.

The second mode (*Convex Hull*), a novel technique, allows users to draw convex shapes directly into thin air by drawing while pinching with one hand (Fig. 9). Technologically, this is achieved by keeping track of all pinch positions and creating a convex hull mesh. As this approach can lead to very sharp corners, which are detrimental to SDF generation, points are smoothed first using a Laplacian filter before being fed into a convex hull generator. This mode works similarly to a lasso metaphor in traditional 2D applications but is limited to convex volumes.

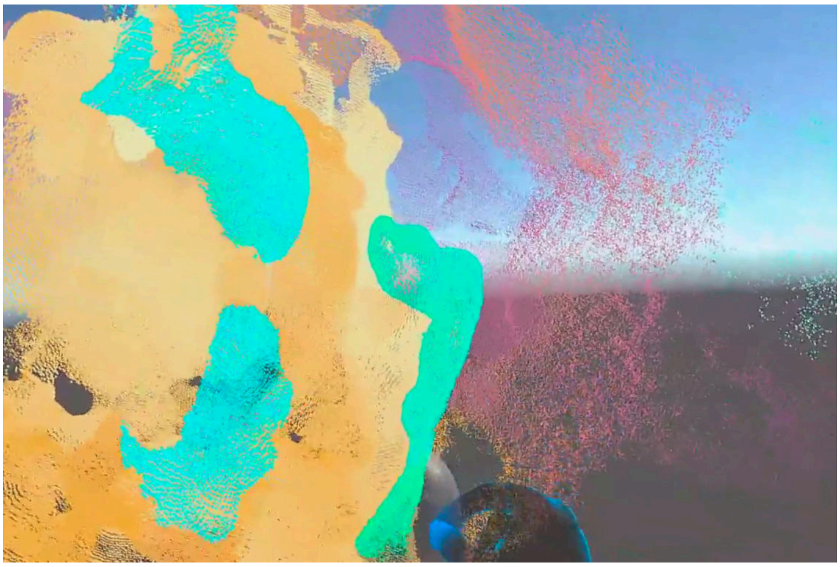


Figure 10: Brushing using a sphere as a selector/brush.

The last two modes allow brushing, which is a continuous selection. In the third mode (*Brush Sphere*), users spawn a sphere the same way as in the first mode and use it directly as a brush (Fig. 10). The sphere can be grabbed to be scaled and translated freely to draw over the point cloud. This mode is also inspired by related work [58, 86, 28]. Technically any volume could be used as a brushing selection volume but for simplification and utility reasons only a sphere was chosen to be used in this mode.

Figure 11: Brushing using the hand itself. Activated by pinching with the other hand.

The fourth mode (*Brush Hands*), a novel approach, allows users to use their hands as a brush directly (Fig. 11). Thanks to the efficient GPU-based SDF calculation, it is possible to recalculate the SDF from the hand mesh in real-time. To avoid the Midas touch problem [90], it is necessary to activate one hand by pinching with the other. This allows for controlled activation of the hand as a selector mesh.

4 Evaluation Methods

This work is evaluated in two ways: with computational performance metrics of the rendering and selection capabilities and a usability study focusing on the usability, accuracy, and effectiveness of the implemented selection techniques.

4.1 Computational Performance

All performance tests were performed on a machine with an Intel Core i9-10900KF CPU and an NVIDIA GeForce RTX 3070 GPU. Unity Version 2022.3.16f1 with DX11 was used, and the prototype application was streamed via Unity's play mode to a Meta Quest Pro with firmware version 67 through Meta Quest Link via Cable.

Execution times of the direct GPU draw call, `Graphics.RenderMeshIndirect()`, are measured on different-sized point clouds ranging from 150′000 to 24 million points. Compute shader execution and render times have been measured using Unity's GPU profiler.

4.2 User Experiment

In order to test the usability of each selection technique against a variety of tasks, as well as to gather overall quantitative and qualitative feedback on the selection approach and answer the following hypothesis, a user experiment was carried out:

- **H1** The use of different selection techniques will significantly affect both the time taken to complete the selection task and the accuracy of the selections.
- **H2** Higher scores on the Mental Rotation Test (MRT) will correlate with faster completion times and greater accuracy in 3D selection tasks.
- **H3** Increased task difficulty will lead to longer completion times and a higher rate of selection errors.
- **H4** Different point clouds have little to no effect on the selection speed and accuracy.

A repeated-measure design with within-subject independent variables, selection technique, task, and point cloud was chosen for the user experiment, where each participant tested all selection techniques on both point clouds and all tasks. This design was chosen to reduce the number of participants needed and increase the study's statistical power. The order of the selection techniques, tasks, task complexity, and point clouds were randomized to mitigate order effects. This resulted in 28 participants x 4 selection techniques x 2 tasks (2 per technique) x 2 point clouds = 448 interactions.

4.2.1 Participants

When comparing the difference between two dependent means (using G*Power [91]), a sample size of 27 participants is required for the user experiment to have 80% power to detect a medium effect size of 0.5 with a significance level of 0.05. Participants were recruited from the computer science institutes at the FHNW, friends, and family. In total, 28 people participated (16 male and 12 female) with an average age of 33 (Mdn = 31, SD = 10). Seventeen pursued higher education levels (9 bachelor's, three master's, and five doctorate degrees), seven people noted high school as their highest finished education level, two people college, and two people a Swiss Certificate of Competence.

In total, 24 people (85.7%) have used VR before, but most (17 people, 60.71%) reported rare to occasional use. Seven of these 24 people use or work with VR monthly to weekly (25%). 18 people (64.3%) reported previous use of hand-tracking technology, 15 of which were rarely to occasional (53.6%) and only three on a monthly to weekly basis (10.7%). 19 out of 24 reported previous interaction with 3D data (e.g., point clouds or 3D models, 67.9%), with 17 reporting academic or professional use (60.71%).

Half of the participants rated their AR/VR/XR experience as basic, while seven rated themselves at intermediate (25%), three participants advanced (10.7%), and four (14.3%) identifying as expert users. In contrast, general computer usage saw higher expertise, with 15 (53.6%) reporting advanced experience and 10 (35.7%) identifying as experts or native users. Only two participants rated themselves at an intermediate level (7.1%) and one at basic level (3.6%). For computer gaming, the participants were more evenly distributed, with ten (35.7%) having advanced experience, nine (32.1%) basic, and five (17.9%) intermediate, while four (14.3%) were experts. This diverse range of expertise among participants provided a solid foundation for evaluating the selection techniques, ensuring that the study's findings are applicable across different levels of user experience.

Two participants reported slightly imperfect color vision, and four mentioned possible motion sickness in VR. However, none of the participants reported any issues regarding color vision during the user test. However, the user test was paused once for a short break for two out of the four participants with possible motion sickness. All participants were able to complete the user test without any further issues.

4.2.2 Materials

The pre-questionnaire collected consent, age, gender, education level, and VR, hand-tracking, and 3D data experience. VR and hand-tracking experience were rated on a 5-point Likert scale from 'rare' to 'daily'. Participants were asked whether they had interacted with 3D data before and, if so, in what context and with what tools. Their proficiency with these tools was rated on a 4-point scale from 'beginner' to 'expert'. A 4-point scale has been chosen over a 5-point scale to avoid a neutral option and get more specific information.

The Mental Rotation Test (MRT) [92] was used to test the participant's spatial abilities. The MRT tests spatial visualization ability and requires participants to rotate objects to match a target object mentally. The test consists of 20 items, and the score is the number of correct answers. The MRT was used to test if there is a correlation between spatial abilities and selection speed, as there has been previous research [93] hinting at a possible correlation. The MRT was conducted before the user test and right after the pre-questionnaire.

Unity Version 2022.3.16f1 with DX11 was used for the user experiment, and the prototype application was streamed via Unity's play mode to a Meta Quest Pro with firmware version 67 through Meta Quest Link via Cable. A Windows laptop with an Intel Core i9-13900H CPU and an NVIDIA GeForce RTX 4070 GPU with 8GB GDDR6 Memory was used to run the Unity application. The use test was conducted in a quiet room with the participant standing in a play area of at least 3x3 meters. The researcher was present during the user test to answer questions and record the results.

For each task and every user, task completion time and selection accuracy were recorded as the primary outcome measures. For this, the Unity application was extended with a `UserTestController` script that essentially wrapped the main `Controller` and performed time and accuracy measurements, handled point cloud switches and displayed and evaluated the current selection task. Additionally, the script exported all user test results as a CSV file for later analysis.

The researcher started time measurement manually when the user was ready to start the task and stopped when the user verbally confirmed that the task was completed or the two-minute time limit was reached. Accuracy was measured by keeping track of how many points of the highlighted region were missing and how many excess points were selected. Compared with the number of points in the highlight and the complete point cloud, a percentage score is calculated by weighting both the highlight accuracy and overall accuracy each at 50%:

$$S = \frac{\left(\frac{E_H}{T_H}\right) + \left(\frac{E}{T}\right)}{2}$$

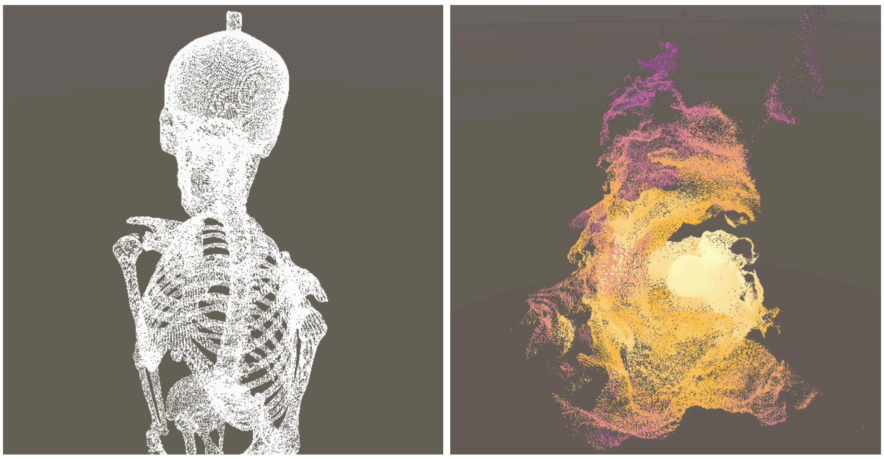
Where E_H is the number of missing points in the task highlight, T_H is the total number of points in the highlight, E is the number of points equal overall, and T is the total number of points in the cloud. S is the final percentage score. The total number of wrongly selected points is also recorded separately by summing up the excess and missing points compared to the required points per task.

Additionally, the number of restarts and selection volumes have been recorded. The number of restarts is used to measure task difficulty, and the number of selection volumes to measure selection technique efficiency. The number of selection volumes is only applicable for the *Shapes* and *Convex Hull* selection modes, as with the brush modes, the number of selectors is always the same. The researcher recorded both values manually during the user test. After each task, participants were asked to rate their confidence in the selection on a 5-point Likert scale.

After the main testing session, participants were asked to complete a post-questionnaire form. This form consisted of the System Usability Scale (SUS) for each selection technique, ratings on the overall experience, hand-tracking, and gestures, all on a 5-point Likert scale. Additionally, qualitative open-text feedback was gathered on what they most liked, found the most challenging, and suggested improvements.

4.2.2.1 Tasks

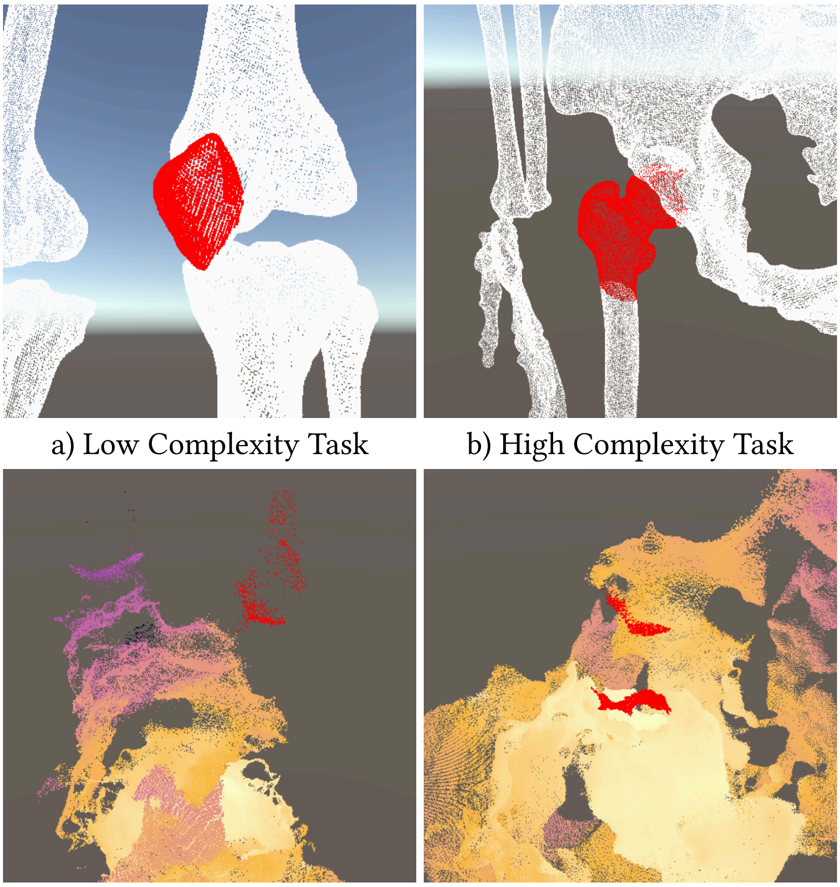
A 3D scan of a skeleton [94] with 1.4 million points, and a point cloud of the local bubble [95, 96] with 786′000 thousand points were chosen as point clouds for the user tests. Both represent data from different fields, biology and astronomy, with different scales and dimensions (Fig. 12).



a) Skeleton b) Local Bubble

Figure 12: Both point clouds are used in the user tests. The skeleton [94] and the local bubble colored by distance [96]

For each point cloud, eight subsets of points have been pre-defined in advance and used as tasks during the user test. The eight tasks are grouped into two equally sized groups based on subjective complexity. Low-complexity tasks are usually task regions where it is straightforward to select the highlight accurately and where other points do not occlude the highlighted volume. High-complexity tasks are split into several smaller highlight groups, are occluded, or their shape requires special care while selecting. These tasks/subsets were manually selected in advance using the prototype of the 3D selection approach, which was based on qualitative judgment and early pilot testing.



a) Low Complexity Task b) High Complexity Task

c) Low Complexity Task d) High Complexity Task

Figure 13: A selection of tasks used in the user study. a) and b) show each a low and high complexity task on the skeleton point scan. c) and d) show both task types on the local bubble point cloud.

During the testing, these tasks were shown to the user as red highlights, i.e., all points inside the task subset were bright red, and all other points were colored in their original point color. In figure 13, examples of tasks for each point cloud and each complexity group are shown.

4.2.3 Procedure

4.2.3.1 Data Collection and Resesarch Design

Participants were sent the pre-questionnaire and the MRT test in advance, and they were informed about the user experiment. They gave their consent to participate in the user test and were informed about the data collection and handling.

Users were briefed on the topic and the goal of the user test. They were then given a tutorial session where they were shown how to use the prototype, what gestures were required, and how the different selection techniques worked. After the tutorial, they were given a training session where they could try out the different selection techniques on a training point cloud for 10 minutes or until they felt confident in using them.

Each selection technique is tested in a randomized trial on two tasks for the main user study. Each task asks the user to select a highlighted number of points as closely as possible. For each point cloud, users iterated through all selection techniques in a random order, and for each selection technique, they performed one task in random order of both the low and high difficulty group. Users were told which selection technique to use and then had to complete two tasks only with that technique. Order effects were mitigated by randomly arranging the order of the selection techniques, tasks, task complexity, and point clouds.

Every task had a two-minute time limit, and users were told regularly how much time they had left. The two-minute time limit was determined based on pilot testing and the assumption that the tasks should be solvable in a reasonable amount of time. If a participant finished the task earlier, they could stop the task via verbal confirmation. After each task, participants were questioned on their confidence level on a 5-point scale. This was repeated for all four selection techniques and eight tasks and then repeated overall for the second point cloud.

Right after the main testing session, participants were asked to complete the post-questionnaire form, which consisted of the System Usability Scale (SUS) for each selection technique, ratings on the overall experience, hand-tracking, and gestures, and qualitative open-text feedback on what they most liked, found the most challenging, and suggestions for improvement.

The overall structure of the user experiment was as follows:

0. (Pre-Questionnaire + MRT)
1. Introduction (5min)
2. Tutorial Session (5min)
3. Training Session (max. 10min)
4. Main Test Sessions (2x 15min)
5. Post-Questionnaire (5min)

Since tasks were time-limited to two minutes, the main testing session could take up to 36 minutes but was completed faster on average. The entire user test took a maximum of 60 minutes per participant.

4.2.3.2 Data Diagnostics and Analysis

Data was first explored and analyzed visually using Tableau [5] and RStudio [97]. Outliers were detected and removed using the IQR (Inter-quartile Range) method, with a scale of 1.5 for outlier detection. Afterwards, the normality assumption was checked using the Shapiro-Wilk test. If the data was not normally distributed, non-parametric tests were chosen in later stages of data analysis.

One-way ANOVA (Analysis of Variance) was used to compare the means across multiple groups and determine whether there were significant effects between the variables. Depending on the ANOVA result, further posthoc tests were carried out—T-tests for normally distributed data and Wilcoxon tests for non-normally distributed data. Effect sizes were calculated using Cohen's d.

5 Results

5.1 Computational Performance

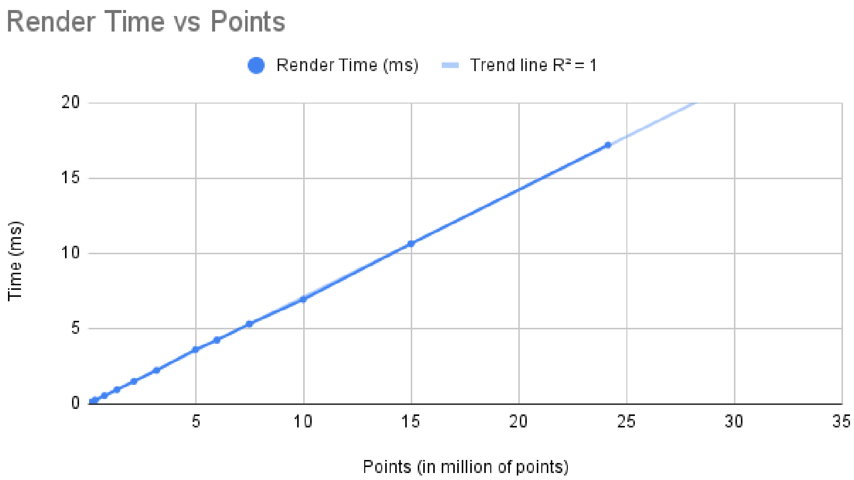


Figure 14: Render call times in ms per point cloud size

The render calls for all points of the point cloud scale perfectly linear (Fig. 14), with all tested point clouds up to 5 million points rendered in under 3.6 ms every frame. The largest point cloud, with 24.1 million points, took, on average, 17.19 ms to render. Points were rendered indirectly and instanced with a direct GPU draw call and measured in Unity's play mode with Unity's GPU profiler.

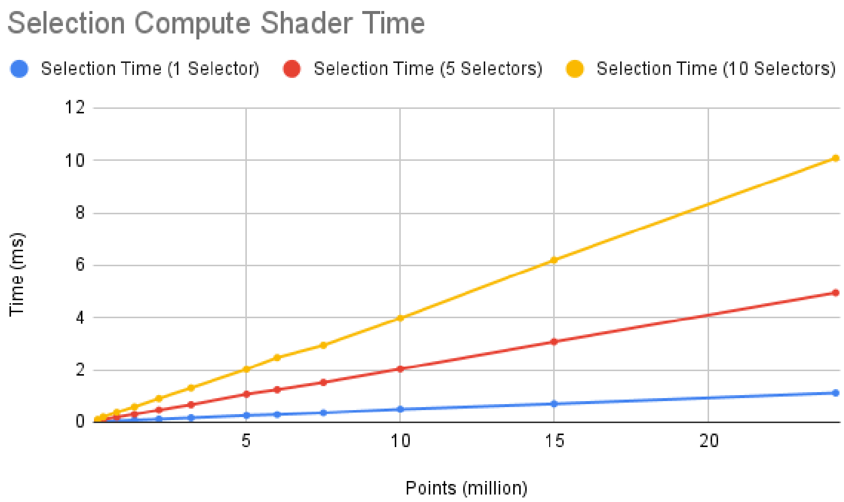


Figure 15: Selection compute shader run times with a varying number of selector shapes in the scene.

The second shader used in the project, the actual selection compute shader, is also measured against the same different-sized point clouds (Fig. 15). This compute shader is dispatched for every selector volume in the scene and effectively compares all points of the point cloud against the SDF texture, storing results in the result compute buffer on the GPU. In figure 15, the linear growth and execution time of the selection compute shader is shown for one, five, and ten selectors, showing perfect $O(n)$ growth with the number of points and $O(n)$ growth with the number of selectors.

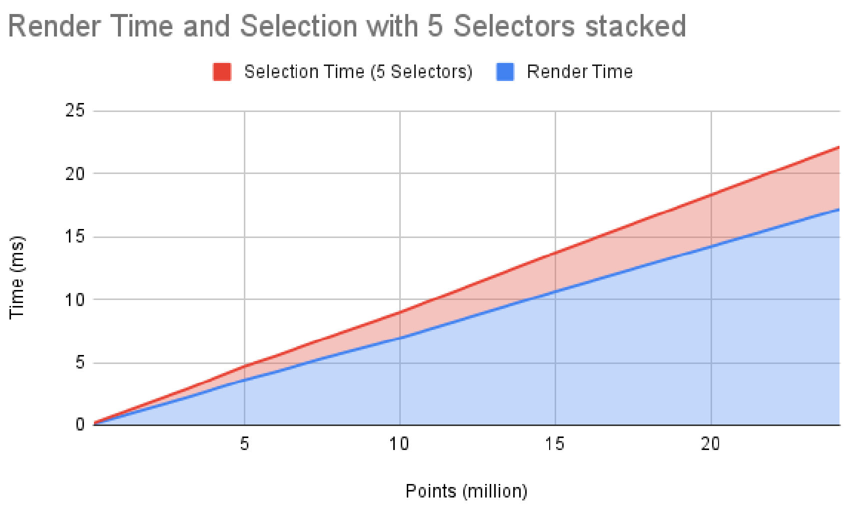


Figure 16: Stacked render times for the indirect instanced GPU draw calls and execution times of the selection compute shader for five selectors.

Figure 16 shows both the render time and compute shader execution time for five selector shapes. The large difference in the time it takes for each shader to run is notable, with the rendering shader taking significantly longer than the selection compute shader alone.

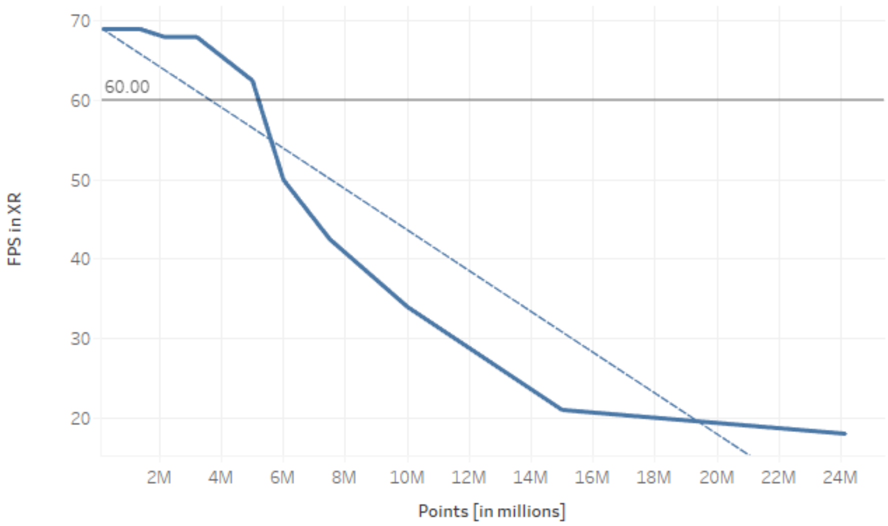


Figure 17: Frame rate on a Meta Quest Pro while selection compute shader is running.

When using an XR headset to display the data in 3D, multi-pass rendering is used, cutting the frame rate in half as one rendering pass is used per eye. The render times in figure 17 are measured while using the application on the headset and while the selection compute shader is running (with five selectors in the scene). These framerates therefore represent the minimum average and are higher when a selector is not moved and triggers a execution of the selection shader. On average this difference was 15-20 frames. As seen in figure 16, the majority of the application's execution time comes from the rendering itself.

Regarding the performance of the SDF baking it was found that this usually took between 1ms to 1.25ms per selector. For most selection techniques an SDF would be baked for the selection volume once at the beginning after dimensions were fix. Even when users transformed the selector afterwards it did not need recalculation of the SDF because the selection shader could calculate the correct position, scale and rotation using the world-to-local matrix of the selector. For the *Brush Hands* mode however, continous recalculation is needed, which added roughly 2ms-2.5ms per frame.

5.2 User Experiment

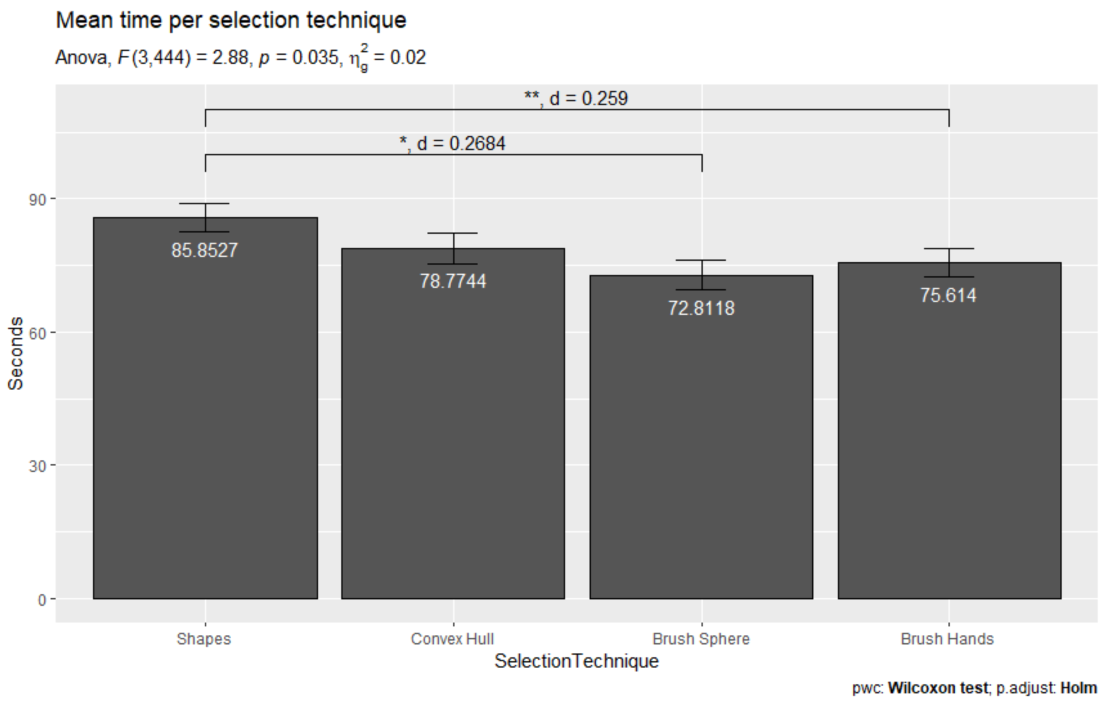


Figure 18: Mean task completion time per selection technique
n=28, total interactions=448 — *** p < .001, ** p < .01, * p < .05, error bars: SEM

Figure 18 shows the average time users completed a task grouped by selection technique. Results suggest that *Brush Sphere* and *Brush Hands* were the fastest techniques and *Shapes* was the slowest. A one-way ANOVA revealed a slightly significant main effect (F(3,444) = 2.88, p = 0.035, η_g^2 = 0.02). A further Wilcoxon test revealed that this effect mainly stems from a significant difference in task completion time between the *Shapes* selection mode and both brushing modes. For all selection techniques, the average selection time lies between 72.8 and 85.9 seconds, with a total average of 78.26 seconds. The third mode, *Brush Sphere*, is the fastest on average, closely followed by the fourth mode, *Brush Hands*, which takes 75.614 seconds. Additionally, it is to be noted that both brushing modes were faster than the other two. Statistically there's only a significant difference between *Shapes* and *Brush Sphere* (p<.05, d=0.2684) and *Shapes* and *Brush Hands* (p<.01, d=0.259). *Convex Hull* had the highest standard deviation compared to all other techniques, and *Brush Hands* had the smallest.

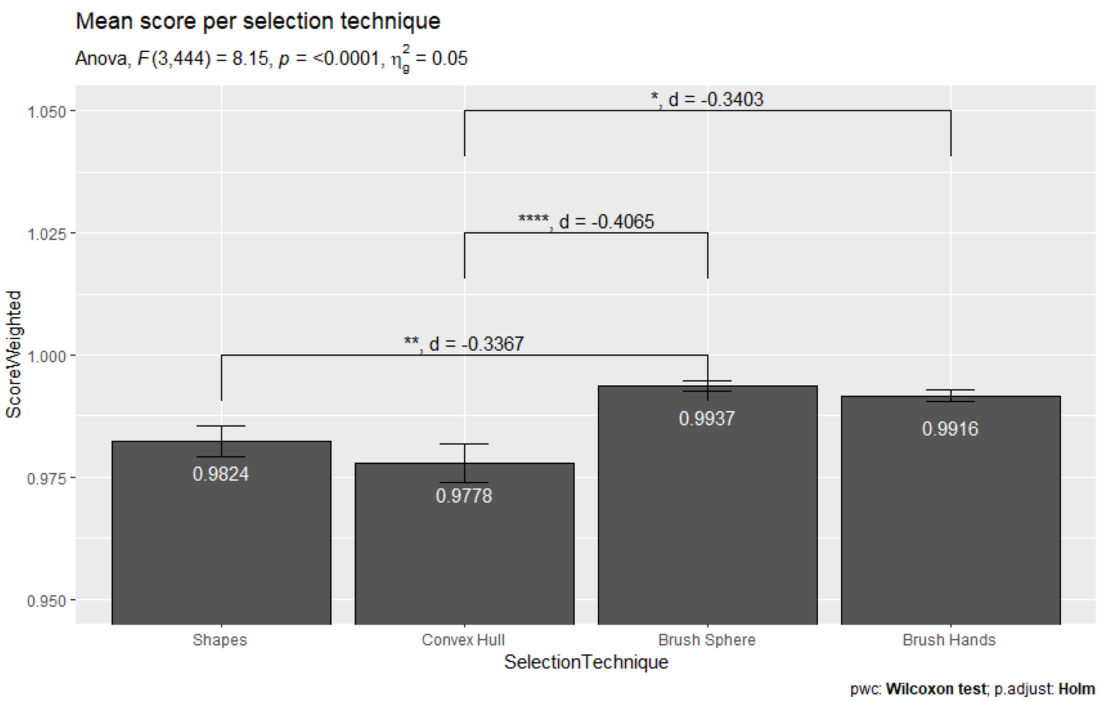


Figure 19: Mean weighted scores grouped by selection technique. Weighted score is a number between 0 and 1 representing percentage of correctly selected points. It is weighted 50/50 between highlight accuracy and total point cloud accuracy (section 4.2.2).
n=28, total interactions=448 — *** p < .001, ** p < .01, * p < .05, error bars: SEM

In figure 19, the average weighted scores per selection mode are shown. Descriptive statistics confirm brushing modes' performance advantage, as seen in Fig. 18. But, in contrast, to mean completion time, *Convex Hull* performed the worst. One-way ANOVA revealed a significant main effect between weighted scores and selection techniques (F(3,444) = 8.15, p = <0.0001, η_g^2 = 0.05). The *Shapes* technique has an average weighted score of 98.242%. This score is relatively high but lower than the scores for *Brush Sphere* and *Brush Hands* but only significantly lower than *Brush Sphere* (p=0.0027, d=0.337). The *Convex Hull* technique has the lowest average weighted score of 97.77% among the four techniques. The *Brush Sphere* technique has the highest average weighted score of 99.370%, next to *Shapes* and significantly higher than *Convex Hull* (p=0.00015, d=0.407). There is no statistical significance between both brushing modes and between both direct selection methods. Overall accuracy scores are very high for each selection technique with a total average 98.64% (weighted score as explained in section 4.2.2).

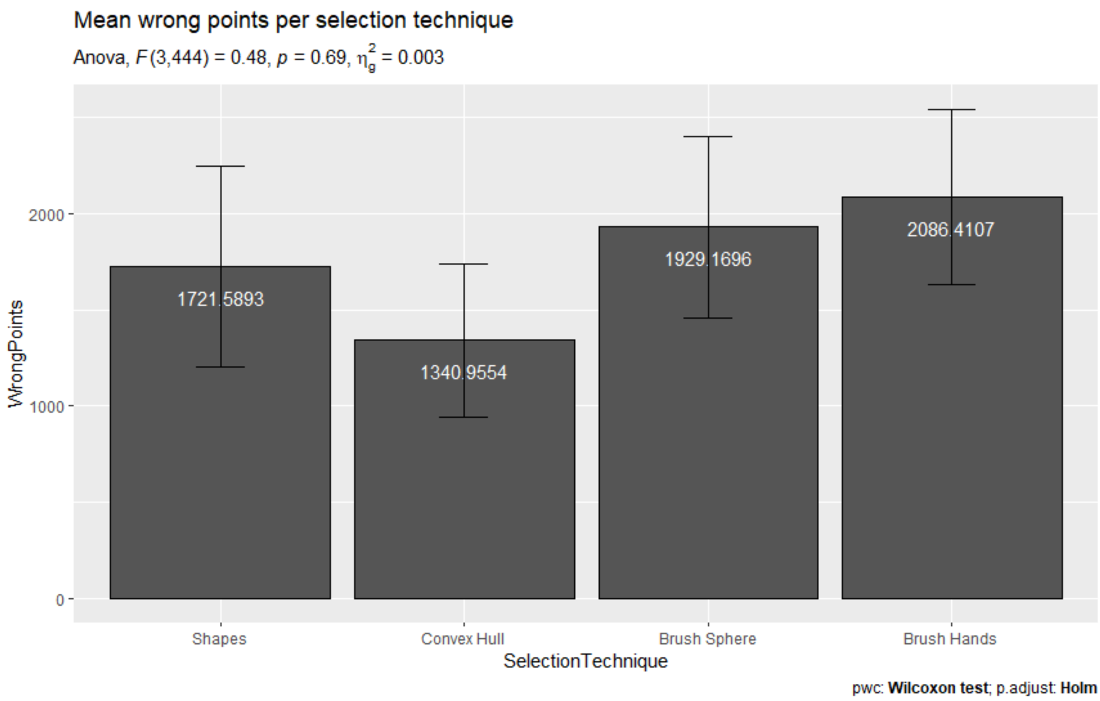


Figure 20: Median wrong points (missing + too much) grouped by selection technique n=28, total interactions=448 — error bars: SEM

Figure 20 shows the average errors per technique. Errors are wrong points, the sum of missing points of the highlighted task subset, and excess points selected. The chart suggests similar performance differences between the selection modes similar to speed (Fig. 18) and accuracy (Fig. 19), yet overall, there is no significant main effect between selection modes and errors.

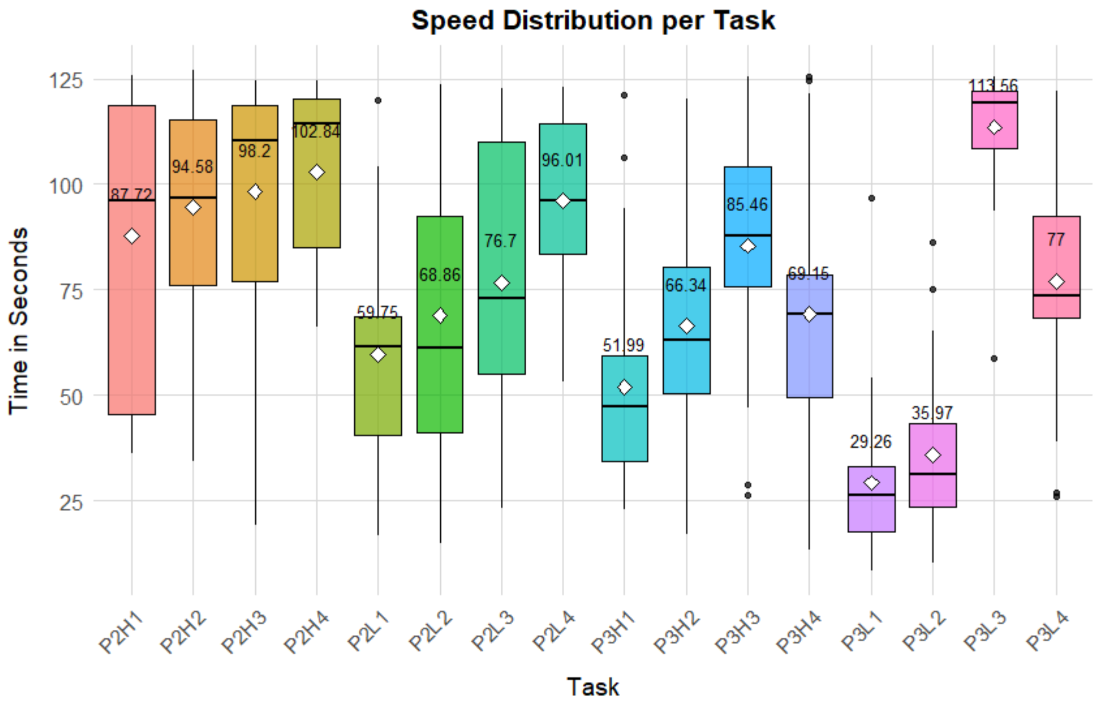


Figure 21: All completion times per Task, n=28, total interactions=448

In figure 21, all completion times for all tasks are shown. Tasks are numbered like `PX[L,H]Y` where X denotes the point cloud number (Skeleton = 2, Local Bubble = 3), L or H denotes whether it is a low or high-complexity task, and Y denotes the task number. Tasks are numbered from one to four in each of the four categories. The chart shows the tasks divided by point cloud on the x-axis, with the high-complexity tasks shown first. The chart suggests that high-complexity tasks require more time to complete than low-complexity tasks. Also notable is the outlier P3L3, which is classified as a low-complexity task yet took the most amount of time on average. Descriptive statistics also suggest that the tasks on the skeleton point cloud take longer on average than on the local bubble point cloud. There is a large variance in completion time between all tasks.

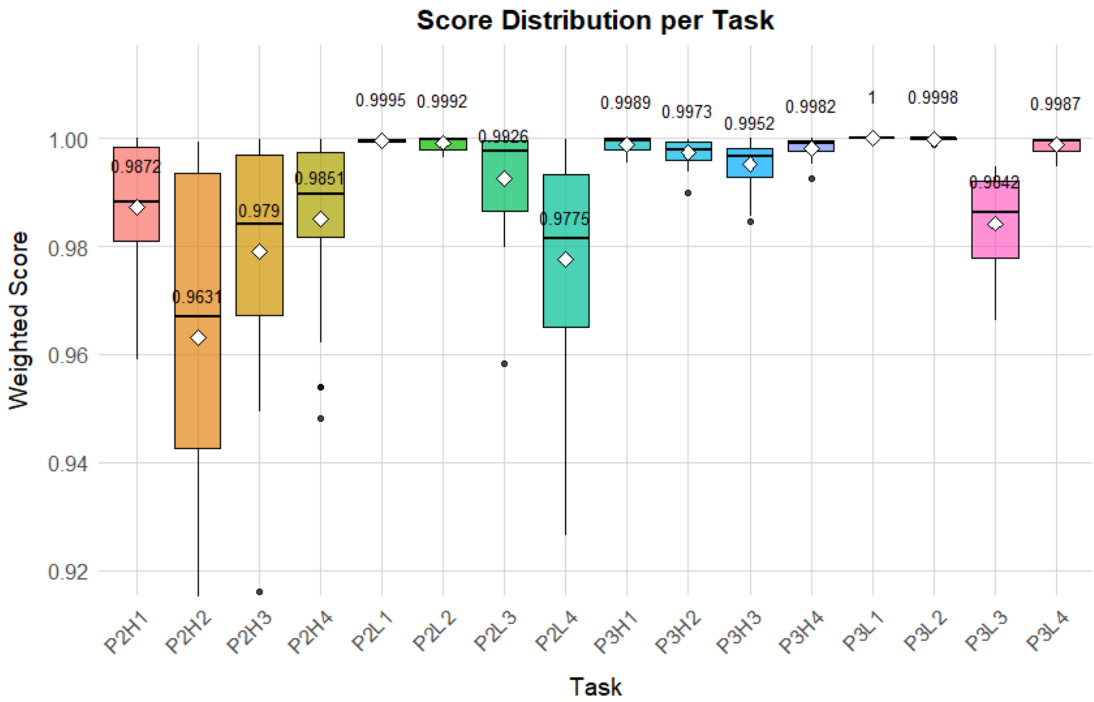


Figure 22: All scores per Task, n=28, total interactions=448

The distribution and mean scores per Task are shown in figure 22. Again, there are noticeable differences between all tasks. The chart suggests again that high-complexity tasks perform worse than low-complexity tasks. Notable is also the difference in mean scores between the skeleton and local bubble point cloud and the difference in variance between both clouds.

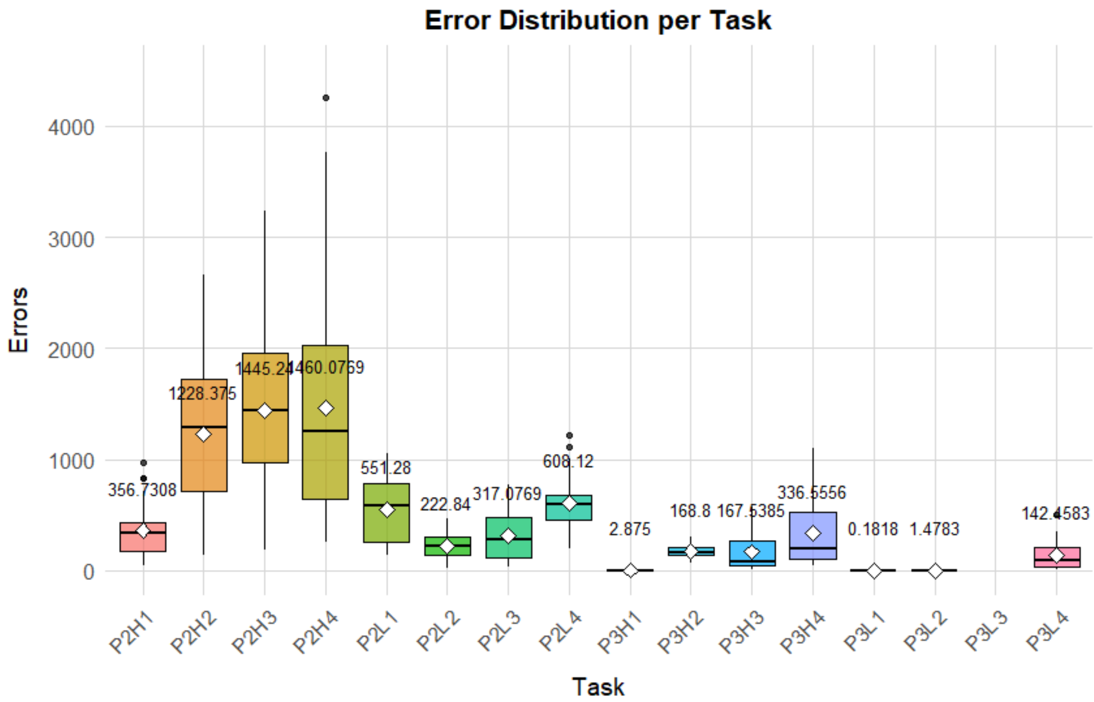
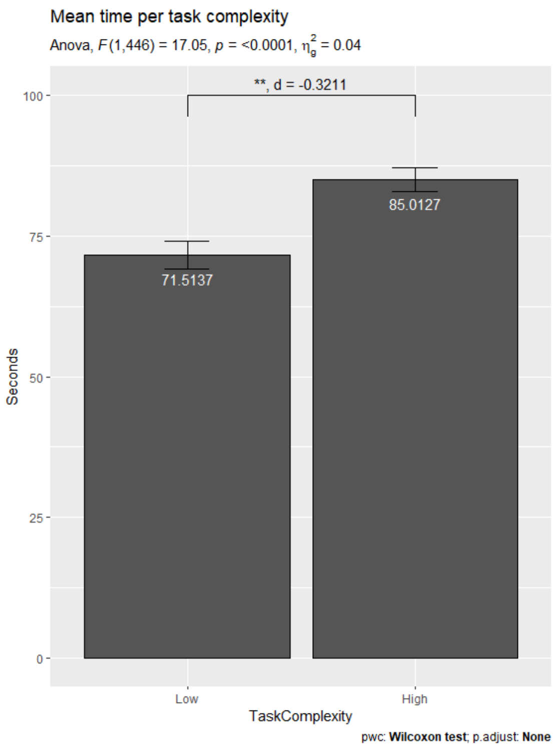
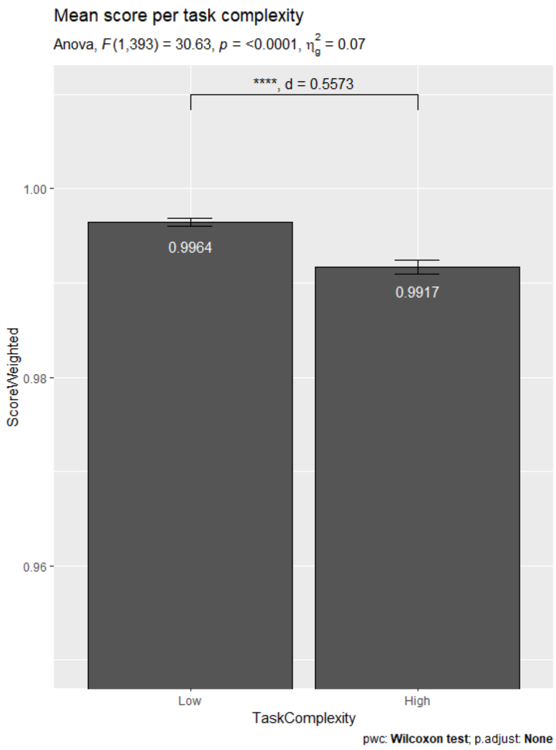


Figure 23: Error distribution per Task. Task P3L3 is not shown as its mean is too large (17′000)

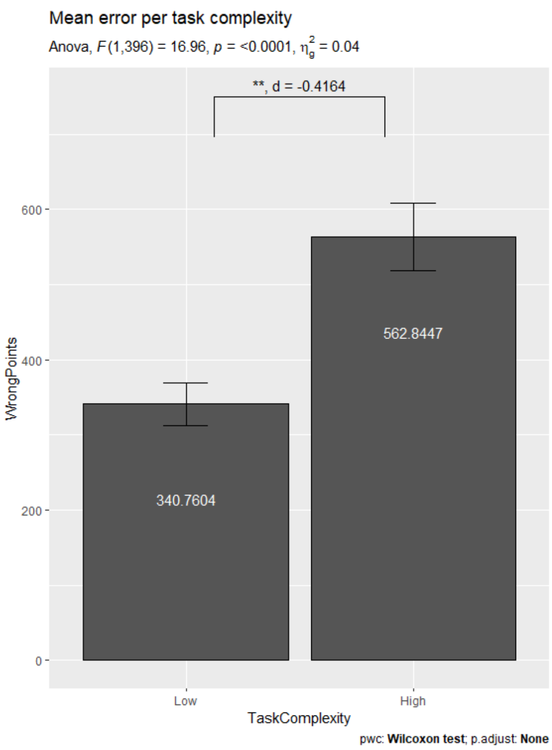
In figure 23, mean errors are shown per Task. Similar to scores in figure 22, a larger number of errors have been made in the Skeleton point cloud. Additionally, the outlier task P3L3 is omitted from view, as it had an average error of 17000 points, far exceeding all other tasks.



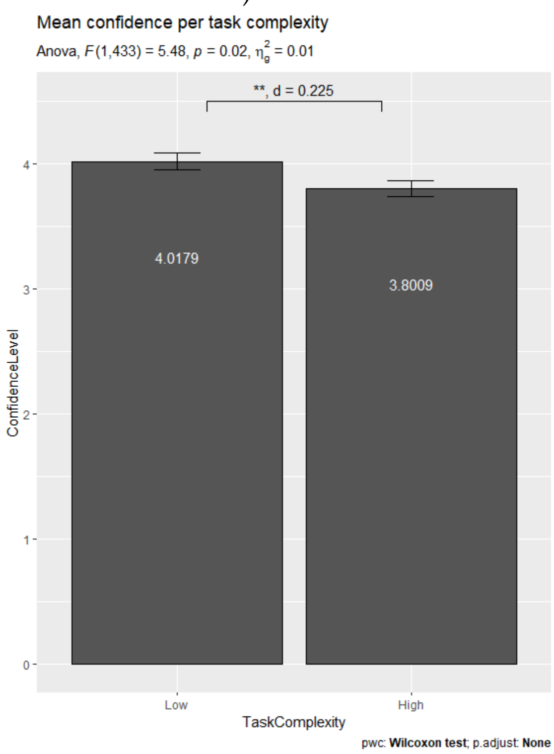
a) Time



b) Score



c) Errors



d) Confidence

Figure 24: Grouped visualizations: a) Mean Speed, b) Mean score, c) Mean error, and d) Mean confidence, all grouped by task complexity (n=28).
*** p < .001, ** p < .01, * p < .05

When task complexity is grouped, as seen in figure 24, significant differences between task complexity on time, score, errors, and confidence become apparent. Easier tasks were actually completed faster on average, with higher average precision and lower wrong points. Participants also rated their selection with a significantly lower complexity Task significantly higher.

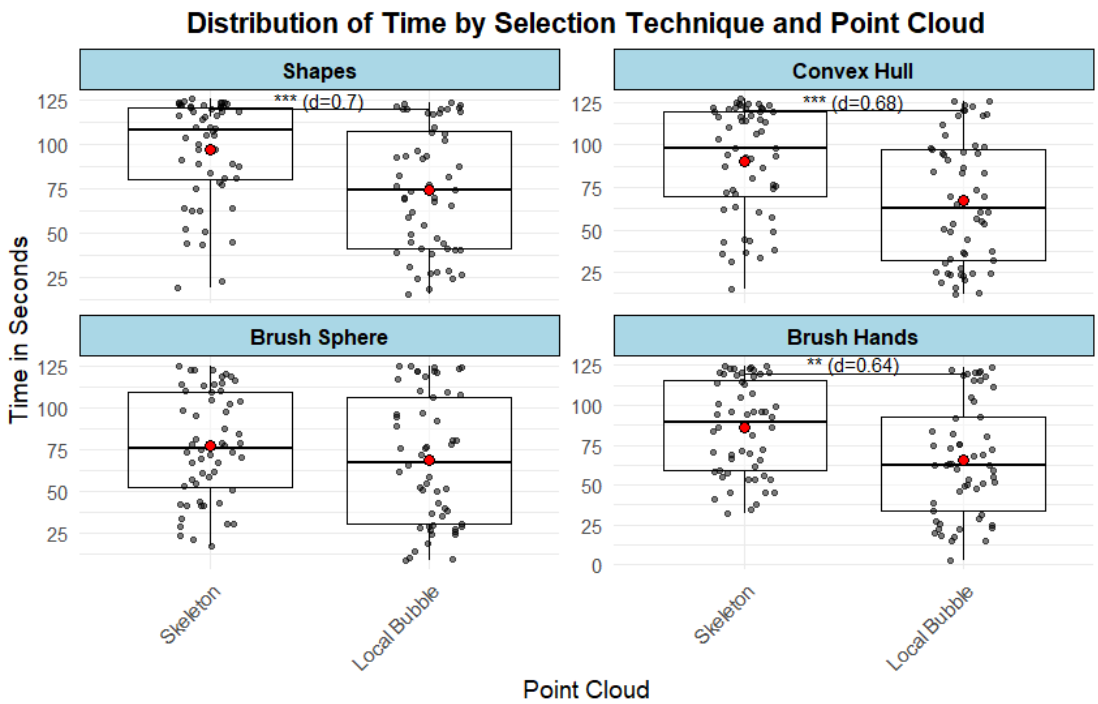


Figure 25: Selection time grouped by selection techniques and point cloud
*** p < .001, ** p < .01, * p < .05

Figure 25 shows a selection time grouped technique, similar to figure 18, but it also groups all by both point clouds. The point cloud has a significant effect on three out of four selection techniques. Only with *Brush Sphere* are there no significant differences.

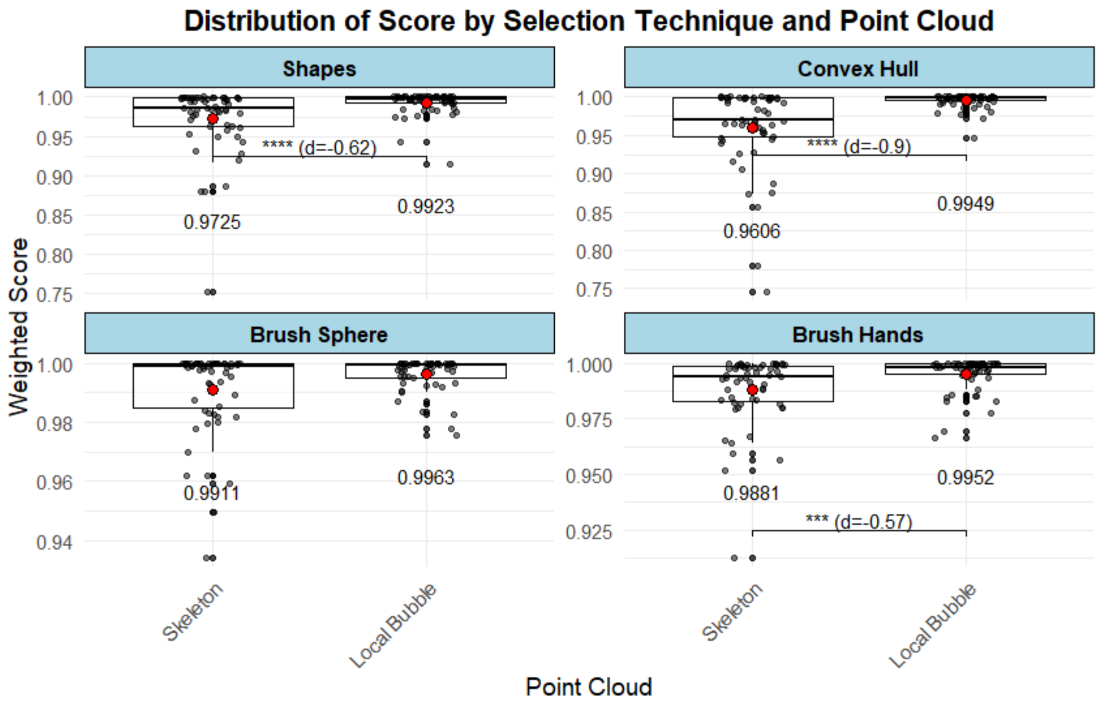


Figure 26: Weighted Scores per Point Cloud and Selection Technique(n=28)
*** p < .001, ** p < .01, * p < .05

The same applies to the weighted score (Fig. 26), where the point cloud significantly affects the weighted score for three out of four selection techniques (all but Brush Sphere). Regarding error (Fig. 27), point clouds have very significant effects on all selection techniques.

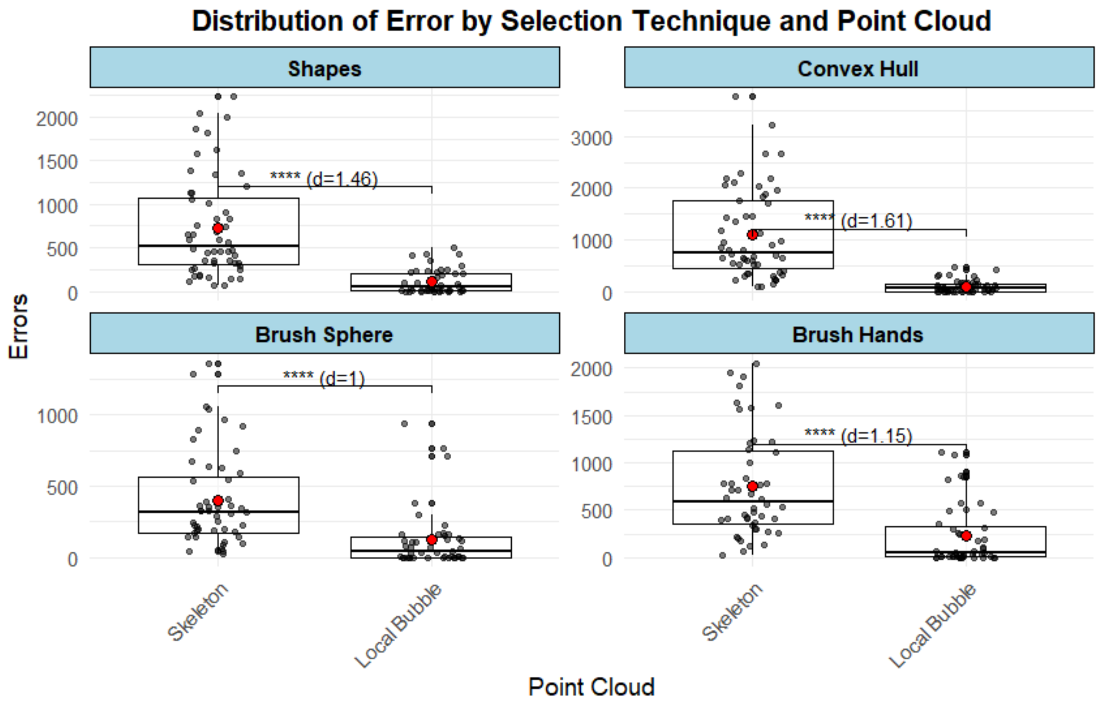
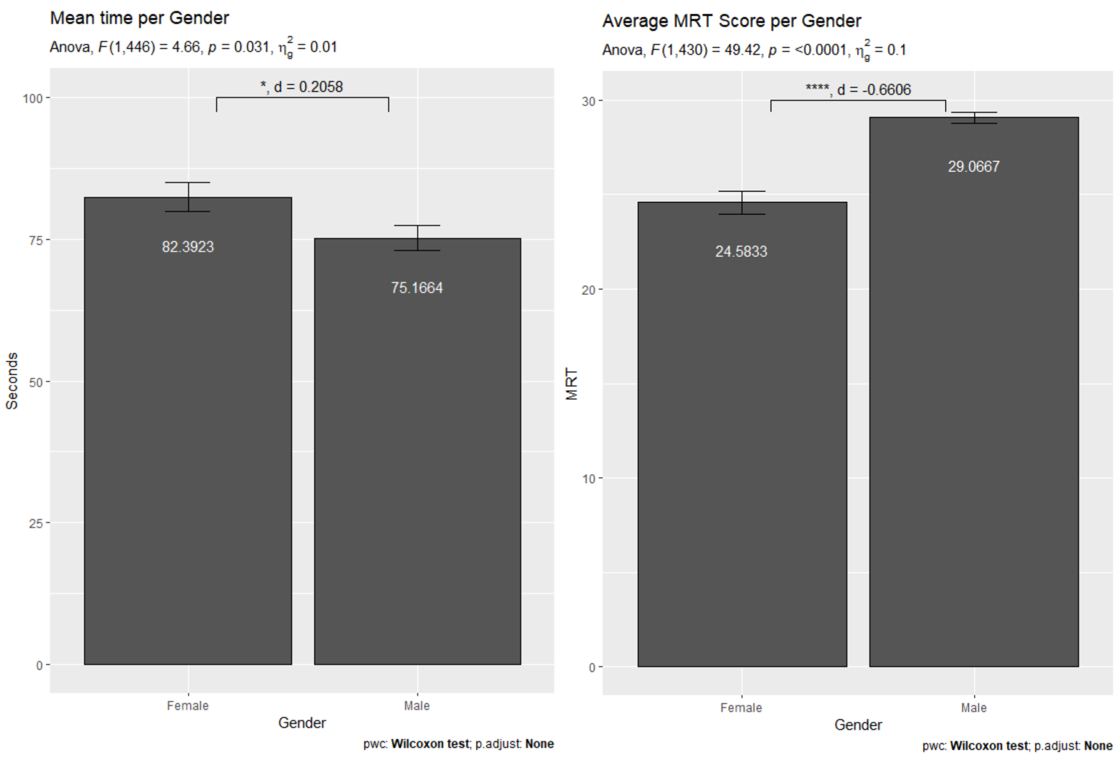


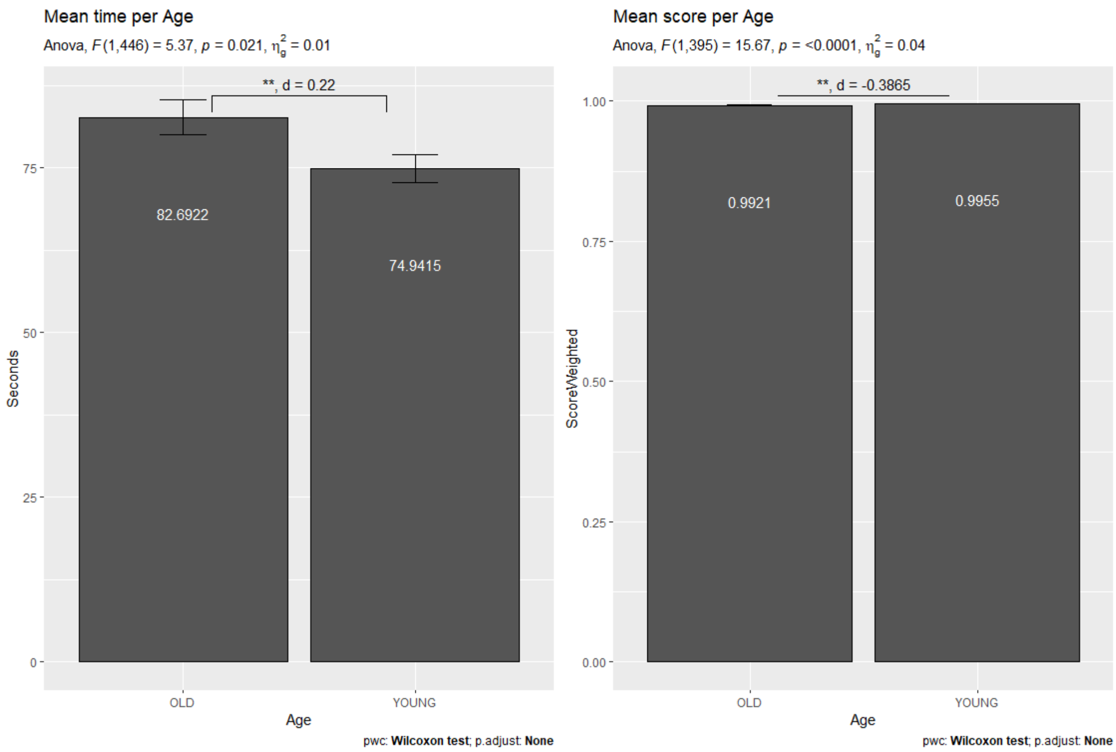
Figure 27: Weighted Scores per Point Cloud and Selection Technique(n=28)
*** p < .001, ** p < .01, * p < .05



a) Time b) MRT score

Figure 28: Comparison of time in seconds and MRT scores grouped by gender.
*** p < .001, ** p < .01, * p < .05

When comparing gender differences (figure 28), only slight significant differences are found in completion time, with women taking slightly longer than men. MRT scores show very significant differences, and men score significantly higher on average. Regarding hypothesis H3, no significant effect of MRT score on task completion or accuracy was found. When analyzing the effect of low and high MRT scores (split at the median of 27 points), no significant effect was found on either selection time (F(1,446) = 0.13, p = 0.71, η_g^2 = 3e-04), score (F(1,401) = 0.23, p = 0.63, η_g^2 = 0.00058) or errors (F(1,394) = 0.05, p = 0.83, η_g^2 = 0.00012). No significant effect of age on MRT score was found (F(1,26) = 0.38, p = 0.54, η_g^2 = 0.01).



a) Time b) Score

Figure 29: Comparison of time and weighted score grouped by age.
*** p < .001, ** p < .01, * p < .05

Comparing age differences, ages are grouped into two groups split by the median age. This results in two bins of 16 young participants and 12 older participants. Only significant differences between both groups in mean selection time and mean weighted score can be found.

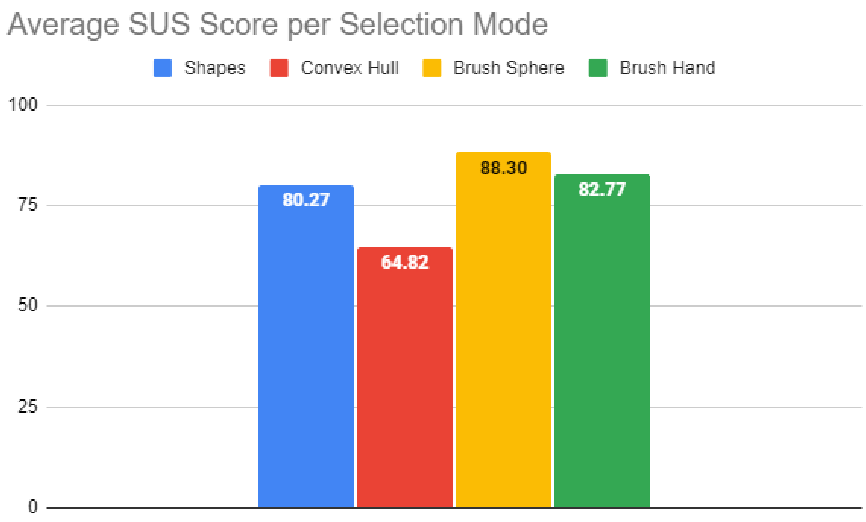
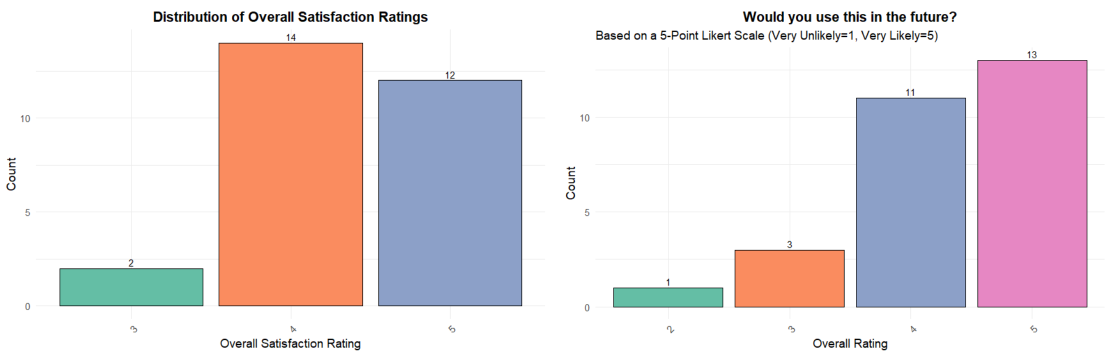


Figure 30: Average SUS Score for each Selection technique. (n=28)

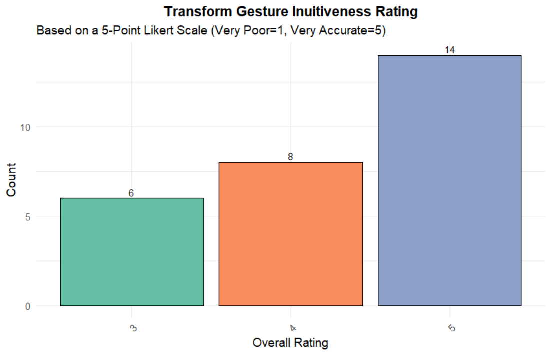
Participants rated the *Brush Sphere* the highest on the SUS scale, closely followed by the *Brush Hands* and then the *Shapes* mode. *Convex Hull* mode was rated slightly below average for SUS scores with 64.82 points. Participants were also asked to rate each technique on a 5-point Likert scale on intuitiveness which resulted in intuitiveness scores being distributed similarly as with the SUS scores. *Brush Sphere* receveived a 4.71 rating (very intuitive), *Brush Hands* 4.32 points (very intuitive), *Shapes* 4.18 points (intuitive) and *Convex Hull* got rated 3.36 (neutral/uncertain) on that scale.



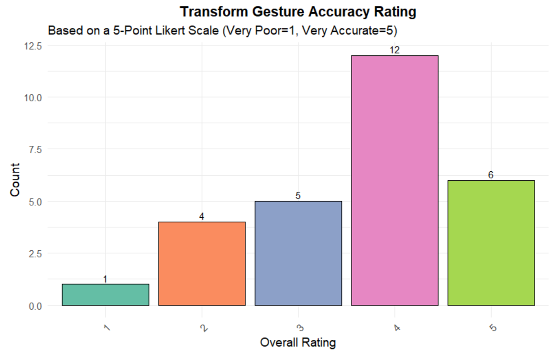
a) Overall Satisfaction Distribution b) Willingness to Use Distribution

Figure 31: Comparison of overall satisfaction rating and willingness to use in future (n=28).

When questioned on their overall satisfaction with the prototype, twelve responded with a rating of 5, 14 with a rating of 4, and only two rated their satisfaction at 3. All ratings were done on a 5-point Likert scale ranging from Very Dissatisfied = 1 to Very Satisfied = 5. Most people would also use this VR prototype for 3D selection in the future, if their work requires them to work with point cloud data. 24 out of 28 participants rated their likeliness use with four or higher. Only four rated their likeliness to use such an application, with 2 or 3.



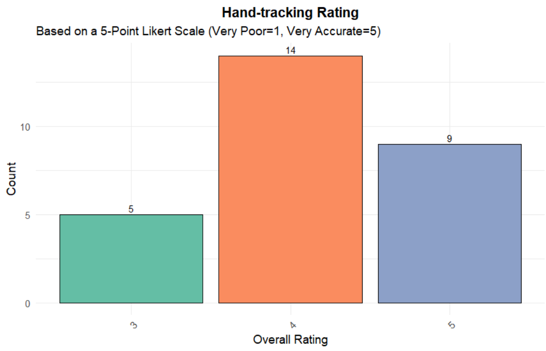
a) Transform Intuitiveness



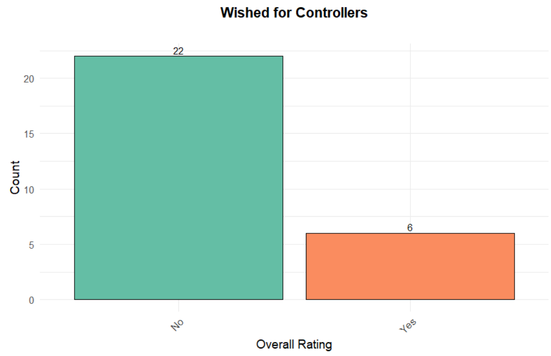
b) Transform Accuracy

Figure 32: Comparison of transform intuitiveness and accuracy ratings (n=28).

Most participants found the transform gesture (double fist gesture) intuitive. They rated its intuitiveness with 4 and 5 on the Likert scale, and six people rated it with a 3. Accuracy was rated more diverse, with six people rating the transform mode's accuracy with a 5, twelve people with a 4, and ten people with a 3 or lower.



a) Hand Tracking Ratings



b) Wish for Controllers

Figure 33: Comparison of hand tracking ratings and users' wish for controllers (n=28)

Regarding hand-tracking accuracy, 23 people rated it four or higher, five rated it 3 on the 5-point Likert scale, and only six out of 28 people wished they had controllers instead of using hand-tracking.

6 Discussion

The proposed 3D selection prototype for point cloud data, including four selection techniques, was implemented and tested on performance, usability, and satisfaction metrics in both a performance evaluation and a user experiment with 28 participants. First, I'll discuss the project's contributions, and then I'll discuss the results of the previous chapter (Chapter 4) and the prototype itself (Chapter 3). Finally, I will discuss limitations and give an outlook on possible future work.

Contributions In this project, I developed a novel 3D selection prototype based upon SDFs as selection volumes capable of querying near-arbitrary volumetric selection regions in real time. The prototype can render and select up to 5 million points (on a test machine) on a tethered VR headset without further optimizations using compute shaders in a GPU-centric approach. Four manual selection techniques have been developed and validated using the prototype. *Shapes* and *Brush Sphere* are based upon existing related work [28, 83–86, 56–58], but *Convex Hull* and *Brush Hands* are novel selection techniques. *Brush Hands* especially highlights the capabilities of the SDF approach, as existing point querying approaches require extensive optimizations, if at all possible, to support real-time changes in the selection volume. However, it is possible to achieve this by combining a fast and real-time SDF generation algorithm with GPU parallelization power. Additionally, a user experiment with 28 participants was carried out, evaluating performance and usability of the selection techniques.

Selection Techniques Results show a significant effect of selection technique on selection speed and weighted scores (accuracy). However, no such effect could be found regarding errors, thereby only confirming hypothesis H1 that selection technique affects time and accuracy for selection time and weighted scores only. Both brushing selection techniques performed better than *Shapes* and *Convex Hull* in terms of speed and accuracy score. These quantitative findings align well with subjective qualitative results from the SUS scores where *Brush Sphere* and *Brush Hands* were rated highest. However, surprisingly, they both resulted in slightly more errors overall, but no statistical significant difference was found in the error count. Over-selection of points is known to happen with mid-air gestures as the user's arm is less stable compare to 2D "tablet" interaction [98] and this effect seems to be stronger in brushing modes compared to direct/immediate selection modes.

Participants rated the *Convex Hull* mode as the lowest on the SUS, but quantitative results only agree regarding accuracy. Despite being disliked by most participants compared to the other selection modes, *Convex Hull* was not significantly slower, nor did it result in significantly more errors. While the idea of the *Convex Hull* mode was to create a 3D volumetric lasso mode, users perceived it as too complicated and unintuitive. Related work on 3D lasso selection techniques[99] showed increased performance for 3D lassos compared to 2D lassos; therefore, similar performance differences were expected for *Convex Hull*, which was not the case. There is no correlation between MRT scores and SUS scores for the *Convex Hull* mode or any other metric. Compared to the other selection modes, *Convex Hull* required higher spatial abilities and probably resulted in a higher mental load, which might partly explain why it performed subjectively worse. Due to the higher mental load, participants concentrated more during this selection mode, resulting in slightly fewer errors. A more comprehensive and in-depth explanation during the introduction could contribute to better subjective scores, and probably, the accuracy of hand-tracking has an effect as well, but all mentioned hypotheses warrant further research.

While users rated the first selection mode, *Shapes*, relatively high on the SUS, it performed worse than both brushing modes but still slightly better than *Convex Hull*. A reason for the overall lower performance of *Shapes* is its flexible selection approach. Users had a free choice between all available volumes and, therefore, could solve a task with as many selector shapes as they liked. Another factor that aligned well with subjective feedback was hand-tracking accuracy, specifically when releasing a shape. Often, the release of a shape results in a slight translation.

All selection techniques except the *Convex Hull* got good average SUS ratings of more than 80, with *Brush Sphere* even with an excellent score of 88.3. *Convex Hull* received a lower but okay score of 64.82. All selection techniques were successful and effective, but the subjective data clearly show that *Convex Hull* needs considerable refinements. The quantitative data shows that all techniques were effective, and qualitative user feedback shows that all modes except *Convex Hull* are intuitive, answering the second research question - *"What are intuitive and effective manual (hand-tracked) selection techniques for the user?"*. Main takeaways of these results are that brushing techniques performed better both quantitatively and qualitatively and that that a 3D lasso technique like *Convex Hull* need considerable improvements.

Compared to previous work [8], the transform mode with which users could translate, scale and rotate the point cloud to their liking, was massively improved and did not suffer from gesture detection problems. However, as the center of rotation and scaling is the center pivot point of the point cloud, the interacton became unnatural for most users as soon as they got very close or inside the point cloud. Users expected the point cloud to zoom from the center point between both hands as well as rotate around that point, which it did no do. While the center pivot point as center of rotation and scaling makes sense when transforming the point cloud from afar, it does require lots of training and feels unnatural when transforming close to the point cloud.

Task Complexity significantly affected all metrics, accepting hypothesis H3. Participants had longer selection times, lower scores, more errors, and lower confidence with high-complexity tasks. I defined tasks manually in advance based on subjective qualitative ratings and pilot results. Apart from one outlier (Task P3L3), results showed that overall tasks were classified correctly, and low-complexity tasks performed better.

Task P3L3 was wrongly classified as a low-complexity task. The task required participants to select half of the local bubble, split through the middle, so roughly 350′000 points. Using *Shapes* with a box shape, *Convex Hull* with a similar box-like shape was expected to solve the task easily. But during the experiment it became clear that many participants did not recognize the inherent box-like structure of the selection volume and just started incrementally selecting as many points as possible. Maybe allowing for more data analysis and exploration in future experiments before selection timing starts could improve selection performance.

Point Clouds / Data Type Contrary to hypothesis H4, point cloud choice, i.e., data type, significantly affected selection performance. Except for *Brush Sphere* with speed and accuracy, participants performed significantly worse with all selection techniques on the Skeleton point cloud. This difference stems partly from the overall shape of the skeleton in combination with the transform mode participants had to use. The transform mode scaled and rotated the point cloud from the point cloud's center point. This transformation makes sense if the viewpoint is far from the point cloud center. Closer or inside the point cloud, this required a a little learning. While users rated the transform mode intuitiveness high, its accuracy was rated slightly lower. The transform modes' usability issues, combined with the elongated shape of the skeleton, might have resulted in overall longer selection times than with the local bubble point cloud.

MRT Unlike pilot results in earlier work [93], no influence of MRT scores on speed, accuracy, or error could be found, rejecting hypothesis H2. MRT scores significantly differed between genders, with males performing significantly better, confirming results from existing literature [100, 101]. There was a slightly significant effect of gender on completion time, where male participants completed selection tasks faster. However, this effect is likely not due to MRT scores but possibly inherent sex differences [102, 103]. Results suggest a slight correlation between MRT score and completion time for male participants, but the sample size is too small for inferential conclusions.

Computational Performance Both the point rendering and the selection compute shader scale with $O(n)$ as they scale with the total number of points and iterate over all points of the point cloud. This can clearly be seen by the perfect linear correlation between execution time and the number of points in section 4.1. Notable, however, is the significant difference in execution time between both shaders, with the instanced indirect rendering pipeline taking significantly longer than the selection calculation of all points. If ten selectors were in the scene, which effectively runs the selection compute shader ten times, the rendering of the points still takes more than twice as long. On an average use case, around five selectors were present at a maximum per selection task, with the average being a lot closer to 1.

While the prototype can handle point clouds of up to 5 million points (on the test machine) without further optimizations, it must be mentioned that the selection algorithm's performance is not the application's bottleneck. The bottleneck is the rendering of the point cloud itself. The selection algorithm is high-speed, thanks to the GPU offloading, and the brute-force approach is more than sufficient for the this use case. The selection algorithm is so fast that it only becomes a problem after around 10 million points (on a test machine), much more than the average point cloud size in the current use case.

SDF baking is done in constant time of 1ms-1.25ms per selector and usually never happens at the same time and only once for each selector as in this use case they are never created at the same time. For the *Brush Hands* mode this means a constant 2ms additional frametime as both hands need live recalculation. But during experiments and tests no performance drawback has been noticed. This brute-force approach and its linear growth become a problem for massive point clouds and extensive optimizations are needed (see section 6.2). Overall, the prototype's performance is good. It showcases the speed of the GPU-based SDF selection algorithm for arbitrary mesh based selection volumes.

Framerates measured on the Meta Quest Pro are around 66fps for point clouds up to 5 million points while the selection compute shader runs. They are roughly 20 frames higher when rendering the point cloud without running the selection compute shader. While a steady framerate of at least 90 fps is recommended for XR applications, the prototype still performs well within the acceptable range and is performing similarly to related work [28]. No issues were observed during the user tests (apart from two people known to suffer from motion sickness). But future iterations of this prototype must address these performance issues.

While related approaches that focused more on optimized rendering of point clouds in Unity have also better selection performance [28] for massive point clouds, they are not capable of selecting points from arbitrary selection volumes like in this work and are limited to spherical or cuboid shapes. The technological approach in this work, validated through a user experiment testing four different selection technique clearly answers the first reseach question - *"How can points inside an arbitrary selection volume be computationally selected and queried in a fast manner?"*. It might not be the fastest solution compared to existing query speeds of tree-like data structures but by attaching the selection process to the render pipeline allows for near-arbitrary selection volumes of even complex meshes.

6.1 Limitations

This work has potential limitations. Participants were recruited mainly from the computer science institutes at the FHNW, but friends and family from varying backgrounds also took part. This allowed for a varied sample size, but this might have been a sample bias due to the large number of participants with prior VR experience (24/28). Additionally, having had friends and family participate allowed for higher sample sizes but also might have introduced a response bias. Both biases could have been mitigated by recruiting a larger and more diverse sample size with more people without VR experience and preferably with less people from friends and family.

Concerning the user experiment, there were technological limitations mainly originating from hand-tracking inaccuracies in the Meta Quest Pro. This headset was chosen for its practicality and availability but, in hindsight, did not have the best hand-tracking on the market. Hand-tracking performance was, in my opinion, still excellent, as most participants self-reported. For future iterations of the project I would suggest using either a Ultraleap based hand-tracking system [82] attached to the tethered headset or future better iterations of Meta headsets with improved hand-tracking.

Data was collected the same way during all user experiments using an editor script in Unity. Accuracy was recorded and calculated automatically and while time was also recorded automatically, it was started manually, potentially leading to inconsistent data collection or errors. Additionally, I instructed all participants and personally showed them how to use the application. This tutorial session was subject to variance and human error and might have resulted in some participants getting a little more explanation than others. The same limitation also applied during testing, where I answered participant's questions. These limitations could have been mitigated by automating the data recording even further and showing a video in the tutorial session for example.

6.2 Future Work

Performance Enhancements Future work must include performance improvements to allow the application to scale and handle massive point clouds. Computational performance results show that the application is usable until a few million points. Therefore, implementing at least a decimated point cloud approach [26, 27] is essential and could guarantee a maximum number of points per frame. This would allow for efficient rendering by reducing the computational complexity while keeping the visual fidelity of the data.

With the current GPU-centric approach, even if render and selection execution times were fine, massive point clouds could not be rendered as they would not fit into GPU memory. An out-of-core octree approach [26, 27] could enable dynamic loading of only necessary data. It is paramount to limit the data transfer between CPU and GPU, so this solution needs careful consideration and optimizations.

Another possible approach to enhancing performance is frustum culling, where only visible points are actually rendered. However, this technique would only partially improve performance when not all points are in view. Therefore, frustum culling is best combined with both techniques mentioned above.

Selection Techniques Advancing brushing techniques is also crucial for improving the precision and intuitiveness of the selection process within point clouds, and future research should focus on refining these methods to enhance user interaction. The current reliance on convex hulls in the second *Convex Hull* selection mode has inherent limitations, especially when dealing with non-convex shapes. Future work should assess whether convex hulls are truly intuitive for users and explore ways to enable the drawing of concave shapes to enhance selection accuracy and allow more flexibility.

Integrating mesh sculpting techniques from established sculpting software could significantly improve selection capabilities. Collaborating with experts or implementing similar systems would allow for more versatile and user-friendly shape generation, increasing the system's overall flexibility. By integrating mesh sculpting techniques from tools like ShapesXR[104], the system could become more powerful and versatile, with enhanced selection capabilities for complex shapes.

While this project only focused on manual selection techniques, it seems sensible and feasable to integrate automated techniques or even techniques leveraging AI. Technically, future AI-powered selection techniques just need to be able to generate a mesh or SDF as a selectin volume in order to integrate such techniques with the proposed SDF based workflow.

The transform mode needs further refinement and a solution needs to be found on how to translate, scale and rotate a point cloud with gestures from within the point cloud. A transform mode massively helps the user for both exploration and correct placement of the point cloud for selection tasks.

This work concentrated on a selection approach using SDFs and evaluated each selection technique separately. But earlier pilot work [8] already showed that users prefer having multiple selection techniques at their disposal. This allows users to freely combine techniques and use the most appropriate mode. User preference and performance on a task where they had free choice over selection modes was emmited from the user experiment for time reasons. Nonetheless, in future versions of this or similar applications it is of paramount importance to present the user with an optimal selection of diverse selection techniques.

Possible other selection techniques and interactions to look into in the future might include: A slicing metaphor similar to [61, 63], either volumetric or gesture based. Due to the ineherent nature of SDFs it would be relatively easy to implement a high-light effect when a selection volumes or hands gets close to a point. This might help with dense point clouds sections and applications similar to [105].

In the current state of the prototype, only direct/immediate and brushing selection can be performed. There is no possibility to combine several selection shapes with boolean operations as it is common in 2D software [3]. SDFs inherently also make boolean operations between them relatively simple and therefore are optimally suited for applications with volumetric boolean operations. Some form of Undo/Redo functionality was also mentioned by users as in the current state, participants had to reset the whole selection when they were not happy with their selection and wanted to restart. Similarly to the last point, allowing for progressive refinement has been shown to increase accuracy over direct/immediate selection techniques [106].

Integrations To expand this project's applicability and reach, it is essential to focus on integrating the system with other visualization and selection software. This integration would involve refining the product to a more polished and finished state and developing pipelines that enable data streaming from external software into the Unity environment in real-time. With such integration, the system could become a versatile tool that complements and enhances existing workflows, making it more valuable to a broader range of users in various fields.

Furthermore, with the rapid advancements in WebGPU and WebXR technologies, there is significant potential to port this system to the web, thereby expanding its accessibility to a broader audience. A web-based version of the system could be particularly beneficial for teams focused on web-based visualization and interaction. Making the system available online could be used in various contexts, from educational settings to collaborative research environments, without the need for specialized software installations. This move towards web integration would pave the way for new opportunities in remote collaboration, data analysis, and selection.

7 Conclusion

This thesis successfully developed and validated a novel approach for 3D point cloud selection in XR environments, leveraging a GPU-centric and SDF-based approach for efficient, real-time selection of arbitrary volumes. The primary objective was to address the limitations of traditional 2D interfaces and existing 3D selection techniques, which often lack precision and are not optimized for complex selection shapes. By implementing and testing various selection techniques within Unity, this research significantly contributes to the field of 3D data interaction.

At the core of this work is the development of a GPU-based SDF selection method, allowing users to interact with and select near-arbitrary subsets of point clouds in real time. This approach proved efficient and scalable, handling up to a few million points without significant performance degradation and optimizations. The research introduced four distinct selection techniques, each offering different interaction paradigms to cater to user preferences and task requirements. These techniques were validated through a comprehensive user experiment involving 28 participants, providing valuable quantitative and qualitative feedback on their effectiveness and usability.

The user study demonstrated the prototype's functionality and offered crucial insights into user preferences and performance. Notably, the results revealed a clear preference for brushing techniques over direct selection modes, with the *Brush Sphere* and *Brush Hands* methods receiving high ratings for their intuitiveness and usability. Additionally, *Convex Hull*, the 3D lasso selection technique, was not well received by the users yet performed only slightly worse compared to the other techniques, warranting future research and improvements. These findings underscore the importance of user-centered design in enhancing the overall experience in XR environments.

Overall, this thesis bridges the gap between traditional 2D methods and modern XR technologies, offering a robust 3D point cloud selection solution. The research demonstrates that it is possible to create intuitive, efficient, and user-friendly systems for interacting with complex 3D data with the right tools and approaches. The contributions made in this work not only advance the state of the art in XR interaction but also open up new possibilities for future developments in this rapidly evolving field.

Statement of Authenticity

I confirm that this master's thesis was written autonomously by me using only the sources, aids, and assistance stated in the report, and that any work adopted from other sources, which was written as part of this thesis, is duly cited and referenced as such.

Brugg-Windisch,

Luca Fluri

Bibliography

[1] A. Fonnet and Y. Prié, "Survey of Immersive Analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2101–2122, Mar. 2021, doi: 10.1109/TVCG.2019.2929033.

[2] B. Bach, R. Dachselt, S. Carpendale, T. Dwyer, C. Collins, and B. Lee, "Immersive Analytics: Exploring Future Interaction and Visualization Technologies for Data Analytics," in *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, in ISS '16. New York, NY, USA: Association for Computing Machinery, Nov. 2016, pp. 529–533. doi: 10.1145/2992154.2996365.

[3] T. Robitaille, C. Beaumont, P. Qian, M. Borkin, and A. Goodman, "glueviz v0.13.1: multidimensional data exploration," *Zenodo*, Feb. 2017, doi: 10.5281/zenodo.1237692.

[4] QGIS Development Team, "QGIS: A Free and Open Source Geographic Information System." [Online]. Available: https://www.qgis.org/

[5] Tableau Software, "Tableau: Business Intelligence and Analytics Software." [Online]. Available: https://www.tableau.com/

[6] J. P. McIntire and K. K. Liggett, "The (possible) utility of stereoscopic 3D displays for information visualization: The good, the bad, and the ugly," in *2014 IEEE VIS International Workshop on 3DVis (3DVis)*, Nov. 2014, pp. 1–9. doi: 10.1109/3DVis.2014.7160093.

[7] J. P. McIntire, P. R. Havig, and E. E. Geiselman, "Stereoscopic 3D displays and human performance: A comprehensive review," *Displays*, vol. 35, no. 1, pp. 18–26, Jan. 2014, doi: 10.1016/j.displa.2013.10.004.

[8] L. Fluri and A. Cöltekin, "3D Data Selection in XR," Sep. 2023.

[9] P. MILGRAM and F. KISHINO, "A Taxonomy of Mixed Reality Visual Displays," Dec. 1994, Accessed: Aug. 17, 2024. [Online]. Available: https://search.ieice.org/bin/summary.php?id=e77-d_12_1321

[10] R. T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, Aug. 1997, doi: 10.1162/pres.1997.6.4.355.

[11] M. Billinghurst, A. Clark, and G. Lee, "A Survey of Augmented Reality," *Foundations and Trends® in Human–Computer Interaction*, vol. 8, no. 2–3, pp. 73–272, Mar. 2015, doi: 10.1561/1100000049.

[12] "A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda - ScienceDirect."

Accessed: Aug. 17, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131519303276

[13] S. Jayaram, J. Vance, R. Gadh, U. Jayaram, and H. Srinivasan, "Assessment of VR Technology and its Applications to Engineering Problems," *Journal of Computing and Information Science in Engineering*, vol. 1, no. 1, pp. 72–83, Jan. 2001, doi: 10.1115/1.1353846.

[14] M. Hmoud, H. Swaity, O. Karram, H. Shibli, S. Swaity, and W. Daher, "High School Students' Engagement in Biology in the Context of XR Technology," *IEEE Access*, vol. 11, pp. 137053–137066, 2023, doi: 10.1109/ACCESS.2023.3338176.

[15] "XR collaboration beyond virtual reality: work in the real world | Journal of Computational Design and Engineering | Oxford Academic." Accessed: Aug. 20, 2024. [Online]. Available: https://academic.oup.com/jcde/article/8/2/756/6175348?login=false

[16] V. Angelov, E. Petkov, G. Shipkovenski, and T. Kalushkov, "Modern Virtual Reality Headsets," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Jun. 2020, pp. 1–5. doi: 10.1109/HORA49412.2020.9152604.

[17] A. Patney *et al.*, "Towards foveated rendering for gaze-tracked virtual reality," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016, doi: 10.1145/2980179.2980246.

[18] R. Albert, A. Patney, D. Luebke, and J. Kim, "Latency Requirements for Foveated Rendering in Virtual Reality," *ACM Trans. Appl. Percept.*, vol. 14, no. 4, pp. 1–13, Sep. 2017, doi: 10.1145/3127589.

[19] G. Buckingham, "Hand Tracking for Immersive Virtual Reality: Opportunities and Challenges," *Frontiers in Virtual Reality*, vol. 2, Oct. 2021, doi: 10.3389/frvir.2021.728461.

[20] C. Mizera, T. Delrieu, V. Weistroffer, C. Andriot, A. Decatoire, and J.-P. Gazeau, "Evaluation of Hand-Tracking Systems in Teleoperation and Virtual Dexterous Manipulation," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1642–1655, Feb. 2020, doi: 10.1109/JSEN.2019.2947612.

[21] "Echtzeit-3D-Entwicklungsplattform und Editor." Accessed: Aug. 20, 2024. [Online]. Available: https://unity.com/products/unity-engine

[22] Epic Games, "Unreal Engine: The Most Powerful Real-Time 3D Creation Tool." [Online]. Available: https://www.unrealengine.com/en-US

[23] "Unity vs Unreal Engine: Pros and Cons [2024 Overview]." Accessed: Aug. 20, 2024. [Online]. Available: https://program-ace.com/blog/unity-vs-unreal/

[24] K. Zhou, Q. Hou, R. Wang, and B. Guo, “Real-time KD-tree construction on graphics hardware,” *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–11, 2008, doi: 10.1145/1409060.1409079.

[25] S. Peters, “Quadtree- and octree-based approach for point data selection in 2D or 3D,” *Annals of GIS*, vol. 19, no. 1, pp. 37–44, Mar. 2013, doi: 10.1080/19475683.2012.758171.

[26] M. Schütz, S. Ohrhallinger, and M. Wimmer, “Fast Out-of-Core Octree Generation for Massive Point Clouds,” *Computer Graphics Forum*, vol. 39, no. 7, pp. 1–13, Nov. 2020, doi: 10.1111/cgf.14134.

[27] E. Neuman-Donihue, M. Jarvis, and Y. Zhu, “FastPoints: A State-of-the-Art Point Cloud Renderer for Unity.” [Online]. Available: https://arxiv.org/abs/2302.05002

[28] J.-P. Virtanen *et al.*, “Interactive dense point clouds in a game engine,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 163, pp. 375–389, May 2020, doi: 10.1016/j.isprsjprs.2020.03.007.

[29] L. Liu, X. Yu, W. Wan, H. Yu, and R. Liu, “Rendering of large-scale 3D terrain point cloud based on out-of-core,” in *2012 International Conference on Audio, Language and Image Processing,* 2012, pp. 740–744. doi: 10.1109/ICALIP.2012.6376712.

[30] OpenVDB Development Team, “OpenVDB: An Open-Source Sparse Volume Data Format.” [Online]. Available: https://www.openvdb.org/

[31] R. Gaugne, Q. Petit, J.-B. Barreau, and V. Gouranton, “Interactive and Immersive Tools for Point Clouds in Archaeology.”

[32] D. Bonatto, S. Rogge, A. Schenkel, R. Ercek, and G. Lafruit, “Explorations for real-time point cloud rendering of natural scenes in virtual reality,” in *2016 International Conference on 3D Imaging (IC3D)*, Dec. 2016, pp. 1–7. doi: 10.1109/IC3D.2016.7823453.

[33] “External memory algorithm.” Accessed: Aug. 20, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=External_memory_algorithm&oldid=1222814748

[34] S. Discher, R. Richter, and J. Döllner, “Concepts and techniques for web-based visualization and processing of massive 3D point clouds with semantics,” *Graphical Models*, vol. 104, p. 101036–101037, Jul. 2019, doi: 10.1016/j.gmod.2019.101036.

[35] I. Quilez, “Inigo Quilez.” Accessed: Aug. 21, 2024. [Online]. Available: https://iquilezles.org/

[36] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and image processing*, vol. 14, no. 3, pp. 227–248, 1980.

[37] J. C. Hart, "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces," *The Visual Computer*, vol. 12, no. 10, pp. 527–545, 1996.

[38] P. Liu, Y. Zhang, H. Wang, M. K. Yip, E. S. Liu, and X. Jin, "Real-time collision detection between general SDFs," *Computer Aided Geometric Design*, vol. 111, p. 102305–102306, Jun. 2024, doi: 10.1016/j.cagd.2024.102305.

[39] P. López-Adeva Fernández-Layos and L. F. S. Merchante, "Convex Body Collision Detection Using the Signed Distance Function," *Computer-Aided Design*, vol. 170, p. 103685–103686, May 2024, doi: 10.1016/j.cad.2024.103685.

[40] X. Zheng, Y. Liu, P. Wang, and X. Tong, "SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation," *Computer Graphics Forum*, vol. 41, no. 5, pp. 52–63, 2022, doi: 10.1111/cgf.14602.

[41] L. Shapira, A. Shamir, and D. Cohen-Or, "Consistent mesh partitioning and skeletonisation using the shape diameter function," *The Visual Computer*, vol. 24, no. 4, pp. 249–259, Apr. 2008, doi: 10.1007/s00371-007-0197-5.

[42] X. Chermain, S. Lucas, B. Sauvage, J.-M. Dischler, and C. Dachsbacher, "Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 4, no. 1, pp. 1–16, Apr. 2021, doi: 10.1145/3451257.

[43] H. H. Söderlund, A. Evans, and T. Akenine-Möller, "Path Tracing of Signed Distance Function Grids," vol. 11, no. 3, 2022.

[44] M. Wimmer, J. Hladuvka, and M. Ilcik, "Proceedings of the 14th Central European Seminar on Computer Graphics."

[45] D. Seyb, A. Jacobson, D. Nowrouzezahrai, and W. Jarosz, "Non-linear sphere tracing for rendering deformed signed distance fields," *ACM Transactions on Graphics*, vol. 38, no. 6, pp. 1–12, Dec. 2019, doi: 10.1145/3355089.3356502.

[46] S. Liu, Y. Zhang, S. Peng, B. Shi, M. Pollefeys, and Z. Cui, "DIST: Rendering Deep Implicit Signed Distance Function With Differentiable Sphere Tracing," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 2016–2025. doi: 10.1109/CVPR42600.2020.00209.

[47] H. Xu and J. Barbič, "Signed Distance Fields for Polygon Soup Meshes," *Graphics Interface 2014*. A K Peters/CRC Press, 2014.

[48] J. Baerentzen, “Robust generation of signed distance fields from triangle meshes,” in *Fourth International Workshop on Volume Graphics, 2005.*, 2005, pp. 167–239. doi: 10.1109/VG.2005.194111.

[49] Y. Wu, J. Man, and Z. Xie, “A double layer method for constructing signed distance fields from triangle meshes,” *Graphical Models*, vol. 76, no. 4, pp. 214–223, Jul. 2014, doi: 10.1016/j.gmod.2014.04.011.

[50] J. Baerentzen and H. Aanaes, “Signed distance computation using the angle weighted pseudonormal,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 3, pp. 243–253, May 2005, doi: 10.1109/TVCG.2005.49.

[51] C. Sigg, R. Peikert, and M. Gross, “Signed distance transform using graphics hardware,” in *IEEE Visualization, 2003. VIS 2003.*, Oct. 2003, pp. 83–90. doi: 10.1109/VISUAL.2003.1250358.

[52] “Unity-Technologies/com.unity.demoteam.mesh-to-sdf.” Accessed: Aug. 18, 2024. [Online]. Available: https://github.com/Unity-Technologies/com.unity.demoteam.mesh-to-sdf

[53] D. Shervheim, “danielshervheim/unity-sdf-generator.” Accessed: Aug. 18, 2024. [Online]. Available: https://github.com/danielshervheim/unity-sdf-generator

[54] Unity, “SDF Bake Tool | Visual Effect Graph | 17.0.3.” [Online]. Available: https://docs.unity3d.com/Packages/com.unity.visualeffectgraph@17.0/manual/sdf-bake-tool.html

[55] R. A. Montano-Murillo, C. Nguyen, R. H. Kazi, S. Subramanian, S. DiVerdi, and D. Martinez-Plasencia, “Slicing-Volume: Hybrid 3D/2D Multi-target Selection Technique for Dense Virtual Environments,” in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Mar. 2020, pp. 53–62. doi: 10.1109/VR46266.2020.00023.

[56] R. Burgess *et al.*, “Selection of Large-Scale 3D Point Cloud Data Using Gesture Recognition,” in *Technological Innovation for Cloud-Based Engineering Systems*, L. M. Camarinha-Matos, T. A. Baldissera, G. Di Orio, and F. Marques, Eds., in IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2015, pp. 188–195. doi: 10.1007/978-3-319-16766-4_20.

[57] F. Homps, Y. Beugin, and R. Vuillemot, “ReViVD: Exploration and Filtering of Trajectories in an Immersive Environment using 3D Shapes,” in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Mar. 2020, pp. 729–737. doi: 10.1109/VR46266.2020.00096.

[58] D. Garrido, R. Rodrigues, A. Augusto Sousa, J. Jacob, and D. Castro Silva, “Point Cloud Interaction and Manipulation in Virtual Reality,” in *2021 5th International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, in AIVR 2021.

New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 15–20. doi: 10.1145/3480433.3480437.

[59] M. Baloup, T. Pietrzak, and G. Casiez, "RayCursor: A 3D Pointing Facilitation Technique based on Raycasting," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, in CHI '19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 1–12. doi: 10.1145/3290605.3300331.

[60] E. Laukka, "Comparing hand tracking and controller-based interactions for a virtual reality learning application." Accessed: Aug. 17, 2024. [Online]. Available: https://oulurepo.oulu.fi/handle/10024/17496

[61] P. Lubos, R. Beimler, M. Lammers, and F. Steinicke, "Touching the Cloud: Bimanual annotation of immersive point clouds," in *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, Mar. 2014, pp. 191–192. doi: 10.1109/3DUI.2014.6798885.

[62] K. A. Siek, Y. Rogers, and K. H. Connelly, "Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs," *Human-Computer Interaction - INTERACT 2005*, vol. 3585. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 267–280, 2005. doi: 10.1007/11555261_24.

[63] F. Bacim, M. Nabiyouni, and D. 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)*, Mar. 2014, pp. 185–186. doi: 10.1109/3DUI.2014.6798882.

[64] M. Sereno, M. Ammi, T. Isenberg, and L. Besançon, "Tangible Brush: Performing 3D Selection with Portable and Position-aware Devices," Oct. 2016. Accessed: Nov. 01, 2022. [Online]. Available: https://hal.inria.fr/hal-01372925

[65] L. Besançon, P. Issartel, M. Ammi, and T. Isenberg, "Hybrid Tactile/Tangible Interaction for 3D Data Exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 881–890, Jan. 2017, doi: 10.1109/TVCG.2016.2599217.

[66] L. Besançon, M. Sereno, L. Yu, M. Ammi, and T. Isenberg, "Hybrid Touch/Tangible Spatial 3D Data Selection," *Computer Graphics Forum*, vol. 38, no. 3, pp. 553–567, Jun. 2019, doi: 10.1111/cgf.13710.

[67] Z. Chen, W. Zeng, Z. Yang, L. Yu, C.-W. Fu, and H. Qu, "LassoNet: Deep Lasso-Selection of 3D Point Clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 195–204, Jan. 2020, doi: 10.1109/TVCG.2019.2934332.

[68] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg, "CAST: Effective and Efficient User Interaction for Context-Aware Selection in 3D Particle Clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 886–895, Jan. 2016, doi: 10.1109/TVCG.2015.2467202.

[69] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg, "Efficient Structure-Aware Selection Techniques for 3D Point Cloud Visualizations with 2DOF Input," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2245–2254, Dec. 2012, doi: 10.1109/TVCG.2012.217.

[70] J. Lalonde, R. Unnikrishnan, N. Vandapel, and M. Hebert, "Scale selection for classification of point-sampled 3D surfaces," in *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*, Jun. 2005, pp. 285–292. doi: 10.1109/3DIM.2005.71.

[71] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." Accessed: Nov. 15, 2022. [Online]. Available: http://arxiv.org/abs/1612.00593

[72] J. J. L. Jr, E. Kruijff, R. P. McMahan, D. Bowman, and I. P. Poupyrev, *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional, 2017.

[73] M. Slater, "Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1535, pp. 3549–3557, Dec. 2009, doi: 10.1098/rstb.2009.0138.

[74] R. P. McMahan, D. A. Bowman, D. J. Zielinski, and R. B. Brady, "Evaluating Display Fidelity and Interaction Fidelity in a Virtual Reality Game," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 626–633, Apr. 2012, doi: 10.1109/TVCG.2012.43.

[75] B. G. Witmer and M. J. Singer, "Measuring Presence in Virtual Environments: A Presence Questionnaire," *Presence: Teleoperators and Virtual Environments*, vol. 7, no. 3, pp. 225–240, Jun. 1998, doi: 10.1162/105474698565686.

[76] J. J. LaViola, "A discussion of cybersickness in virtual environments," *SIGCHI Bull.*, vol. 32, no. 1, pp. 47–56, Jan. 2000, doi: 10.1145/333329.333344.

[77] J. F. Dumas and J. C. Redish, *A Practical Guide to Usability Testing*. USA: Greenwood Publishing Group Inc., 1993.

[78] J. Bailenson, *Experience on Demand: What Virtual Reality Is, How It Works, and What It Can Do*. W. W. Norton & Company, 2018.

[79] J. Linietsky and A. Manzur, "Godot Engine - Free and open source 2D and 3D game engine." Accessed: Jul. 30, 2024. [Online]. Available: https://godotengine.org/

[80] C. Khundam, V. Vorachart, P. Preeyawongsakul, W. Hosap, and F. Noël, "A Comparative Study of Interaction Time and Usability of Using Controllers and Hand

Tracking in Virtual Reality Training," *Informatics*, vol. 8, no. 3, p. 60–61, Sep. 2021, doi: 10.3390/informatics8030060.

[81] S. Kapsoritakis, "A comparative study of virtual reality hand-tracking and controllers." Accessed: Aug. 20, 2024. [Online]. Available: http://www.theseus.fi/handle/10024/754933

[82] "Leap Motion Controller 2 - Ultraleap." Accessed: Aug. 21, 2024. [Online]. Available: https://leap2.ultraleap.com/products/leap-motion-controller-2/

[83] A. Hamelin and E. Dubois, "Design and evaluation of an interaction technique for volume selection in a 3D point cloud," in *Proceedings of the 27th Conference on l'Interaction Homme-Machine*, in IHM '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1–10. doi: 10.1145/2820619.2820622.

[84] A. Olwal, H. Benko, and S. Feiner, "SenseShapes: using statistical geometry for object selection in a multimodal augmented reality," in *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, Oct. 2003, pp. 300–301. doi: 10.1109/ISMAR.2003.1240730.

[85] H. Benko and S. Feiner, "Balloon Selection: A Multi-Finger Technique for Accurate Low-Fatigue 3D Selection," in *2007 IEEE Symposium on 3D User Interfaces*, Mar. 2007. doi: 10.1109/3DUI.2007.340778.

[86] D. Akers, A. Sherbondy, R. Mackenzie, R. Dougherty, and B. Wandell, "800Exploration of the brain's white matter pathways with dynamic queries," in *IEEE Visualization 2004*, Oct. 2004, pp. 377–384. doi: 10.1109/VISUAL.2004.30.

[87] "PLY - Polygon File Format." Accessed: Aug. 21, 2024. [Online]. Available: https://paulbourke.net/dataformats/ply/

[88] K. Takahashi, "keijiro/Pcx." Accessed: Aug. 21, 2024. [Online]. Available: https://github.com/keijiro/Pcx

[89] "Comma-separated values." Accessed: Aug. 21, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Comma-separated_values&oldid=1237715653

[90] T. Conte, "The Midas Touch effect: the most unknown phenomenon in UX design." Accessed: Aug. 20, 2024. [Online]. Available: https://uxdesign.cc/the-midas-touch-effect-the-most-unknown-phenomenon-in-ux-design-36827204edd

[91] "Universität Düsseldorf: G*Power." Accessed: Aug. 14, 2024. [Online]. Available: https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower

[92] R. N. Shepard and J. Metzler, "Mental Rotation of Three-Dimensional Objects," *Science*, vol. 171, no. 3972, pp. 701–703, Feb. 1971, doi: 10.1126/science.171.3972.701.

[93] A. Ambühl, L. Fluri, C. Zucker, A. Goodman, and A. Cöltekin, "Effectiveness and perceived usefulness of a handheld ar cube for examining 3d spatial structures," in *SpaceCHI Workshop at the CHI2022*, 2022.

[94] Artec3D, "Human skeleton HD | Free 3D model | Professional 3D scanning solutions." Accessed: Aug. 05, 2024. [Online]. Available: https://www.artec3d.com/3d-models/human-skeleton-hd

[95] T. J. O'Neill, C. Zucker, A. A. Goodman, and G. Edenhofer, "The Local Bubble is a Local Chimney: A New Model from 3D Dust Mapping." Accessed: Jul. 29, 2024. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2024arXiv240304961O

[96] T. O'Neill, "Local Bubble is a Local Chimney." Accessed: Jul. 29, 2024. [Online]. Available: https://theo-oneill.github.io/localbubble/

[97] "Posit." Accessed: Aug. 15, 2024. [Online]. Available: https://www.posit.co/

[98] R. A. Montano-Murillo, C. Nguyen, R. H. Kazi, S. Subramanian, S. DiVerdi, and D. Martinez-Plasencia, "Slicing-Volume: Hybrid 3D/2D Multi-target Selection Technique for Dense Virtual Environments," in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Mar. 2020, pp. 53–62. doi: 10.1109/VR46266.2020.00023.

[99] W. Zhou, S. Correia, and D. H. Laidlaw, "Haptics-Assisted 3D Lasso Drawing for Tracts-of-interest Selection in DTI Visualization."

[100] M. Hegarty, "Ability and sex differences in spatial thinking: What does the mental rotation test really measure?," *Psychonomic Bulletin & Review*, vol. 25, no. 3, pp. 1212–1219, Jun. 2018, doi: 10.3758/s13423-017-1347-z.

[101] M. S. Masters and B. Sanders, "Is the gender difference in mental rotation disappearing?," *Behavior Genetics*, vol. 23, no. 4, pp. 337–341, Jul. 1993, doi: 10.1007/BF01067434.

[102] Z. Estes and S. Felker, "Confidence Mediates the Sex Difference in Mental Rotation Performance," *Archives of Sexual Behavior*, vol. 41, no. 3, pp. 557–570, Jun. 2012, doi: 10.1007/s10508-011-9875-5.

[103] R. Ariel, N. A. Lembeck, S. Moffat, and C. Hertzog, "Are there sex differences in confidence and metacognitive monitoring accuracy for everyday, academic, and psychometrically measured spatial ability?," *Intelligence*, vol. 70, pp. 42–51, Sep. 2018, doi: 10.1016/j.intell.2018.08.001.

[104] "ShapesXR — Spatial design and prototyping platform for teams." Accessed: Aug. 21, 2024. [Online]. Available: https://www.shapesxr.com/

[105] A. Prouzeau, M. Cordeil, C. Robin, B. Ens, B. H. Thomas, and T. Dwyer, "Scaptics and Highlight-Planes: Immersive Interaction Techniques for Finding Occluded Features in 3D Scatterplots," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, in CHI '19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 1–12. doi: 10.1145/3290605.3300555.

[106] R. Kopper, F. Bacim, and D. A. Bowman, "Rapid and accurate 3D selection by progressive refinement," in *2011 IEEE Symposium on 3D User Interfaces (3DUI)*, Mar. 2011, pp. 67–74. doi: 10.1109/3DUI.2011.5759219.

Appendix

Source Code

All code for the Unity application is hosted on the local FHNW Gitlab server at this address:

https://gitlab.fhnw.ch/iit/mse/luca_fluri_p7-p9_2022-2024/p9_hand-tracked_3d_data_selection_of_point_clouds_in_xr_2024

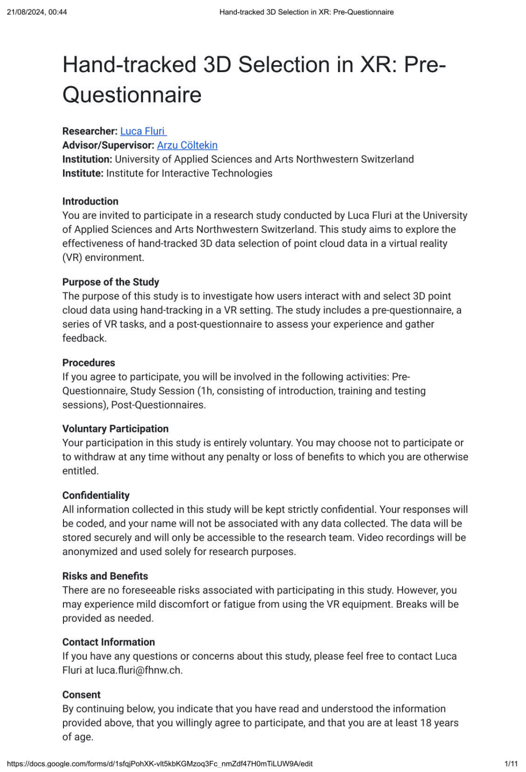
In case you have no access to repository, please contact me at luca.fluri@fhnw.ch

Evaluation

Results

Raw data and results from the user experiment are available upon request from luca.fluri@fhnw.ch

Pre-Questionnaire



21/08/2024, 00:44 Hand-tracked 3D Selection in XR: Pre-Questionnaire

Hand-tracked 3D Selection in XR: Pre-Questionnaire

Researcher: Luca Fluri
Advisor/Supervisor: Arzu Cöltekin
Institution: University of Applied Sciences and Arts Northwestern Switzerland
Institute: Institute for Interactive Technologies

Introduction
You are invited to participate in a research study conducted by Luca Fluri at the University of Applied Sciences and Arts Northwestern Switzerland. This study aims to explore the effectiveness of hand-tracked 3D data selection of point cloud data in a virtual reality (VR) environment.

Purpose of the Study
The purpose of this study is to investigate how users interact with and select 3D point cloud data using hand-tracking in a VR setting. The study includes a pre-questionnaire, a series of VR tasks, and a post-questionnaire to assess your experience and gather feedback.

Procedures
If you agree to participate, you will be involved in the following activities: Pre-Questionnaire, Study Session (1h, consisting of introduction, training and testing sessions), Post-Questionnaires.

Voluntary Participation
Your participation in this study is entirely voluntary. You may choose not to participate or to withdraw at any time without any penalty or loss of benefits to which you are otherwise entitled.

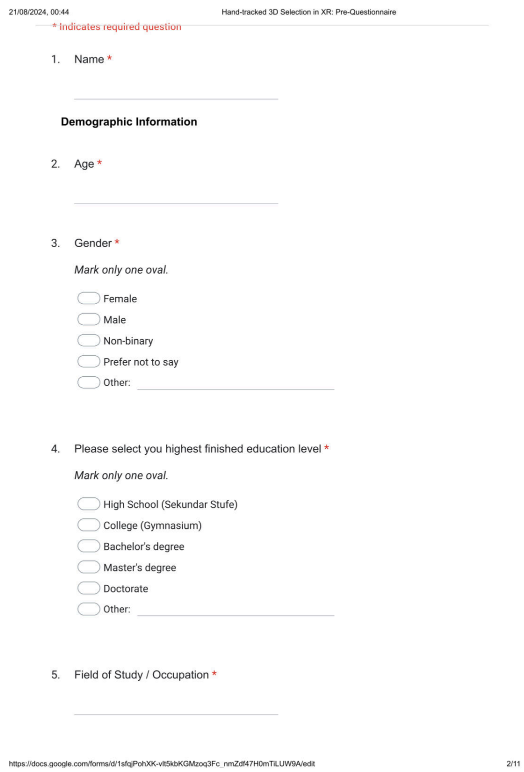
Confidentiality
All information collected in this study will be kept strictly confidential. Your responses will be coded, and your name will not be associated with any data collected. The data will be stored securely and will only be accessible to the research team. Video recordings will be anonymized and used solely for research purposes.

Risks and Benefits
There are no foreseeable risks associated with participating in this study. However, you may experience mild discomfort or fatigue from using the VR equipment. Breaks will be provided as needed.

Contact Information
If you have any questions or concerns about this study, please feel free to contact Luca Fluri at luca.fluri@fhnw.ch.

Consent
By continuing below, you indicate that you have read and understood the information provided above, that you willingly agree to participate, and that you are at least 18 years of age.

https://docs.google.com/forms/d/1sfqjPohXK-vlt5kbKGMzoq3Fc_nmZdf47H0mTiLUW9A/edit 1/11



21/08/2024, 00:44 Hand-tracked 3D Selection in XR: Pre-Questionnaire

* Indicates required question

1. Name *

Demographic Information

2. Age *

3. Gender *

Mark only one oval.

- Female
- Male
- Non-binary
- Prefer not to say
- Other:

4. Please select you highest finished education level *

Mark only one oval.

- High School (Sekundar Stufe)
- College (Gymnasium)
- Bachelor's degree
- Master's degree
- Doctorate
- Other:

5. Field of Study / Occupation *

https://docs.google.com/forms/d/1sfqjPohXK-vlt5kbKGMzoq3Fc_nmZdf47H0mTiLUW9A/edit 2/11

Experience with VR (Virtual-Reality)

6. Have you used VR before? *

Mark only one oval.

- Yes
- No

7. If yes, how frequently do you use VR?

Mark only one oval.

- Daily
- Weekly
- Monthly
- Occasionally
- Rarely

8. If yes, What VR systems have you used most?

Mark only one oval.

- Oculus/Meta (Rift, Quest...)
- HTC Vive
- Playstation VR
- Pico
- Other: ____________

Experience with Hand-Tracking

9. Have you used hand-tracking technology before? *

Mark only one oval.

- Yes
- No

10. If yes, how frequently do you use hand-tracking?

Mark only one oval.

- Daily
- Weekly
- Monthly
- Occasionally
- Rarely

11. If yes, what hand-tracking systems have you used most?

Mark only one oval.

- Leapmotion/Ultraleap
- Oculus/Meta
- Apple Vision Pro
- Other: ____________

Familiarity with 3D Data

12. Please rate your level of experience in using: *

Mark only one oval per row.

	Basics	Intermediate	Advanced	Expert/Native
Computer in General	◯	◯	◯	◯
Computer Games	◯	◯	◯	◯
AR/VR/XR	◯	◯	◯	◯

13. Have you interacted with 3D data (e.g. point clouds, 3D models) before? *

Mark only one oval.

- Yes *Skip to question 14*
- No *Skip to question 18*

Familiarity with 3D Data - 3D Data

14. In what context have you used 3D Data? *

Mark only one oval.

- Academic
- Professional
- Hobbyist
- Other: ____________

15. Please describe the type of 3D data you have worked with. *

16. What software or tools have you used to work with 3D data? *

17. How would you rate your proficiency with these tools? *

Mark only one oval.

- Beginner
- Intermediate
- Advanced
- Expert

Health and Safety

18. Do you have any conditions that may be affected by using VR? (e.g., motion sickness, epilepsy) *

Mark only one oval.

- Yes
- No

19. If yes, please specify

20. Have you ever been told by a professional that you have **imperfect color vision**? *

Mark only one oval.

- Yes
- No

21. Have you ever been told by a professional that you have **imperfect stereo vision**? *

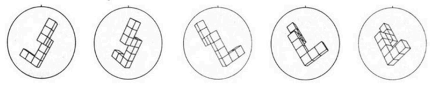
Mark only one oval.

- Yes
- No

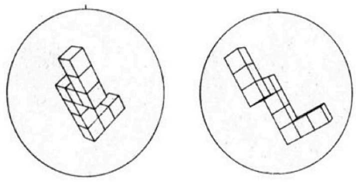
22. Do you have any additional comments or concerns regarding your participation in this study?

MRT Explanation

This is a test of a person's ability to look at a drawing of a given object and find the same object within a set of dissimilar objects. The only difference between the original object and the chosen object will be that they are presented at different angles. An illustration of this principle is given below where the same single object is given in five different positions. Look at each of them to satisfy yourself that they are only presented at different angles from one another.

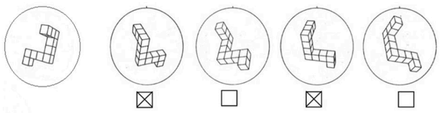


Here are two drawings of new objects. They cannot be made to match the above five drawings. Satisfy yourself that they are different from the above.



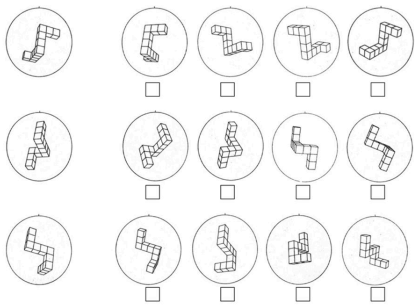
Practice Problem

Now let's do some sample problems. For each problem there is a primary object on the far left. You are to determine which two of four objects to the right are the same object given on the far left. In each problem always two of the four drawings are the same object as the one on the left. You are to put Xs in the boxes below the correct ones, and leave the incorrect ones blank. The first sample problem is done for you.



Do the rest of the sample problems shown here by yourself. Which two drawings of the four on the right show the same object as the one on the left? There are always two correct answers for each problem. In the experiment you will be able to mark the two correct drawings.

Answers: (1) first and second drawings are correct | (2) first and third drawings are correct | (3) second and third drawings are correct



Please do the MRT Test

Please open the following link (MRT Survey) in a new tab and do the MRT. But please don't forget to submit this form after you've finished it.
https://forms.office.com/e/UjnkCTpQfy

Thank you

Thanks a lot for your time and effort in filling out this questionnaire.

If you have any questions, please feel free to ask me anytime.
Luca Fluri (luca.fluri@fhnw.ch)

Post-Questionnaire

21/08/2024, 00:44 Hand-tracked 3D Selection in XR: Post-Questionnaire

Hand-tracked 3D Selection in XR: Post-Questionnaire

* Indicates required question

1. Name *

SUS

for each selection mode.

2. 1. I think that I would like to use this system frequently *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)					
Mode 2 (Convex Drawing)					
Mode 3 (Brush Sphere)					
Mode 4 (Brush Hand)					

https://docs.google.com/forms/d/1tW5fOoJzxnntHH5K0nB15vbSw4Wbj8_-kw_Wks4uScM/edit 1/9

21/08/2024, 00:44 Hand-tracked 3D Selection in XR: Post-Questionnaire

3. 2. I found the system unnecessarily complex *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)					
Mode 2 (Convex Drawing)					
Mode 3 (Brush Sphere)					
Mode 4 (Brush Hand)					

4. 3. I thought the system was easy to use *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)					
Mode 2 (Convex Drawing)					
Mode 3 (Brush Sphere)					
Mode 4 (Brush Hand)					

https://docs.google.com/forms/d/1tW5fOoJzxnntHH5K0nB15vbSw4Wbj8_-kw_Wks4uScM/edit 2/9

21/08/2024, 00:44 Hand-tracked 3D Selection in XR: Post-Questionnaire

5. 4. I think that I would need the support of a technical person to be able to use this system *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)					
Mode 2 (Convex Drawing)					
Mode 3 (Brush Sphere)					
Mode 4 (Brush Hand)					

6. 5. I found the various functions in this system were well integrated *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)					
Mode 2 (Convex Drawing)					
Mode 3 (Brush Sphere)					
Mode 4 (Brush Hand)					

https://docs.google.com/forms/d/1tW5fOoJzxnntHH5K0nB15vbSw4Wbj8_-kw_Wks4uScM/edit 3/9

21/08/2024, 00:44 Hand-tracked 3D Selection in XR: Post-Questionnaire

7. 6. I thought there was too much inconsistency in this system *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)					
Mode 2 (Convex Drawing)					
Mode 3 (Brush Sphere)					
Mode 4 (Brush Hand)					

8. 7. I would imagine that most people would learn to use this system very quickly *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)					
Mode 2 (Convex Drawing)					
Mode 3 (Brush Sphere)					
Mode 4 (Brush Hand)					

https://docs.google.com/forms/d/1tW5fOoJzxnntHH5K0nB15vbSw4Wbj8_-kw_Wks4uScM/edit 4/9

9. 8. I found the system very cumbersome to use *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)	◯	◯	◯	◯	◯
Mode 2 (Convex Drawing)	◯	◯	◯	◯	◯
Mode 3 (Brush Sphere)	◯	◯	◯	◯	◯
Mode 4 (Brush Hand)	◯	◯	◯	◯	◯

10. 9. I felt very confident using the system *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)	◯	◯	◯	◯	◯
Mode 2 (Convex Drawing)	◯	◯	◯	◯	◯
Mode 3 (Brush Sphere)	◯	◯	◯	◯	◯
Mode 4 (Brush Hand)	◯	◯	◯	◯	◯

11. 10. I needed to learn a lot of things before I could get going with this system *

Mark only one oval per row.

	1 (Strongly Disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly Agree)
Mode 1 (Shapes)	◯	◯	◯	◯	◯
Mode 2 (Convex Drawing)	◯	◯	◯	◯	◯
Mode 3 (Brush Sphere)	◯	◯	◯	◯	◯
Mode 4 (Brush Hand)	◯	◯	◯	◯	◯

Overall Feedback

12. Overall, how satisfied are you with using hand-tracked 3D selection in VR? *

Mark only one oval.

	1	2	3	4	5	
Very	◯	◯	◯	◯	◯	Very Satisfied

13. Imagine that your profession requires you to work with point cloud data: How likely are you to use this VR system for 3D data selection tasks in the future? *

Mark only one oval.

	1	2	3	4	5	
Very	◯	◯	◯	◯	◯	Very Likely

14. How intuitive did you find the hand-tracked 3D selection tools? *

Mark only one oval per row.

	1 (Not intuitive)	2	3 (Neutral)	4	5 (Very intuitive)
Mode 1 (Shapes)	◯	◯	◯	◯	◯
Mode 2 (Convex Drawing)	◯	◯	◯	◯	◯
Mode 3 (Brush Sphere)	◯	◯	◯	◯	◯
Mode 4 (Brush Hand)	◯	◯	◯	◯	◯

15. How immersive did you find the VR environment? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very immersive

16. How intuitive did you find the pointcloud transformations (fist gesture)? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very intuitive

17. How accurate did you find the pointcloud transformations (fist gesture)? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very accurate

18. What did you like most about using hand-tracked 3D selection in VR? *

____

____

____

____

____

19. What did you find most challenging about using the hand-tracked 3D selection tools? *

____

____

____

____

____

20. Do you have any suggestions for improving the hand-tracked 3D selection and manipulation tools in VR? *

____

____

____

____

____

21. What was your impression of the hand-tracking? *

Mark only one oval.

	1	2	3	4	5	
Very	◯	◯	◯	◯	◯	Very Good

22. Did you ever wish to have controllers instead? *

Mark only one oval.

◯ Yes
◯ No

23. Any additional comments or feedback?