Fachhochschul-Masterstudiengang
HUMAN-CENTERED COMPUTING
4232 Hagenberg, Austria

Volumetric Data Interaction in AR Using a Handheld Touch-Sensitive Tablet

Master's thesis

to obtain the academic degree
Master of Science in Engineering

Submitted by

Janine Denise Mayer B.Sc.

Supervisor: FH-Prof. Dr. Christoph Anthes M.Sc.

September 2022

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere. This printed copy is identical to the submitted electronic version.

Hagenberg, September 29, 2022

Janine Denise Mayer, BSc

Contents

Declaration iii

Preface vii

Abstract viii

Kurzfassung ix

1 Introduction 1
- 1.1 Research Questions . . . 2
- 1.2 Overview . . . 2

2 Related Work 3
- 2.1 Closely Related Work . . . 3
  - 2.1.1 Physical Props as Handheld Devices . . . 3
  - 2.1.2 Handheld Devices with Computing Power . . . 5
- 2.2 Input Devices . . . 8
  - 2.2.1 Generic Input Devices . . . 9
  - 2.2.2 Panels and Tablets . . . 12
- 2.3 Virtual Displays . . . 14
- 2.4 Interaction Techniques . . . 16
  - 2.4.1 Mid-Air Gestures . . . 16
  - 2.4.2 Touch and Spatial Input . . . 17
  - 2.4.3 Best Practices . . . 18
- 2.5 Interaction Tasks . . . 21
  - 2.5.1 Selection . . . 21
  - 2.5.2 Manipulation . . . 23

3 Concept and Design 25
- 3.1 Requirements . . . 25
  - 3.1.1 Non-Goals . . . 26
- 3.2 Architecture . . . 26
- 3.3 Use Cases . . . 27
- 3.4 Volumetric Data Sets . . . 28
  - 3.4.1 Computer Tomography . . . 28
  - 3.4.2 Rendering Techniques . . . 29

3.4.3 Prototype Data Sets . . . 29
3.5 Interaction . . . 30
3.5.1 Navigation . . . 32
3.5.2 Spatial Transformation . . . 32
3.5.3 Selection . . . 33
3.5.4 Clipping . . . 34
3.5.5 Snapshots . . . 36
3.6 User Interface . . . 38
3.6.1 Main Menu . . . 39
3.6.2 Selection Mode . . . 40
3.6.3 Exploration Mode . . . 41

4 Implementation **43**
4.1 Technology . . . 43
4.1.1 Hardware . . . 43
4.1.2 Software . . . 46
4.2 Data . . . 46
4.2.1 Volumetric Data . . . 47
4.2.2 Surface Data . . . 49
4.3 Prototypical Implementation . . . 51
4.3.1 Network Communication . . . 52
4.3.2 User Input . . . 55
4.3.3 Prototype States . . . 57
4.3.4 Data Exploration . . . 63
4.4 Limitations . . . 72
4.4.1 Hardware . . . 72
4.4.2 Network . . . 72
4.4.3 Touch based input . . . 72
4.4.4 Frozen cutting plane . . . 73
4.4.5 Intersection orientation . . . 73
4.4.6 Cutting plane position within model . . . 73
4.4.7 Calculation format . . . 74

5 Evaluation **75**
5.1 User Study . . . 75
5.1.1 Design . . . 75
5.1.2 Environment . . . 77
5.1.3 Participants . . . 78
5.1.4 UEQ Results . . . 78
5.1.5 Individual Feedback . . . 82
5.2 Discussion . . . 87
5.2.1 Suggestions . . . 88

6 Conclusion **89**
6.1 Outlook . . . 90

A List of Acronyms **91**

B Evaluation Documents **92**

References **104**
Literature 104

Preface

This master thesis was written and rewritten over the course of more than one year of research, sketching, developing, and evaluating. At first, I was able to research this topic for 20 hours a week as part of the HIVE research group, working on the X-PRO project under the guidance of my supervising professor Dr. Anthes. The related work and concept chapter were concluded during a semester abroad, when I was staying at the Iwate Prefectural University (IPU) in Japan. The remaining chapters were finished during my subsequent travel through Tokyo, Kyoto, and Osaka. Here at my final destination in South Korea, I gave the thesis a last polish before handing it in. Now that the scientific content of this work is completed, I dedicate this preface to thanking all the people who have supported me directly or indirectly in the completion of this work. First, and most importantly, I want to thank professor Dr. Anthes who has helped me a lot with my research and has always given great advice and feedback on my writing. He was an insightful lead throughout the whole creation process of these pages and even encouraged me to hand in a workshop paper on my research, which has been published last November at the Interactive Surfaces and Spaces 2021[1]. Also, a big thank you to all my colleagues at the X-PRO research team: Judith, Daniel, Fabian, Stefan, and Andreas, who were a great team, fun to be around and helped me with ideas, explanations, and advice for the development of my prototype as well as the design of the user study, and the process of writing a paper or thesis. During my stay at the IPU, I spent most of my time at the research lab of professor Dr. Prima, who provided hardware for me to finish and evaluate the prototype. I am very greatful for his help and the support of the people in my lab. I enjoyed the time I could spend with you and thank you very much for helping me out with my prototype and evaluation: Yuta, Ray, Jihad, Kenta, Syahid, and Chaitali. Thank you also to Gemma, for keeping up with my 1-am-prototype-stress-soup-cooking and Bernhard for not killing me after waking him past midnight to borrow a cable. Of course, I also thank my friends Sandra, Linda, Felix, David, and Jackie, who were there to correct my writing and make great suggestions. This thesis has around 700 comments worked into it, without which these pages would not look the way they do now. I also owe a thank you to the translation website *DeepL*, which helped out with formulations when I was at a loss of words and the online LaTex editor *Overleaf* for hosting and compiling all these pages. To close this preface, I thank my family and friends who gave me space when I did not want to talk about the progress of this master thesis and helped me to keep my mind off of it and enjoy the time I did not spend brooding over these pages.

[1] http://kops.uni-konstanz.de/handle/123456789/55462

Abstract

Since Augmented Reality (AR) was first mentioned in the 1990s [4], the domain has evolved greatly in terms of available technologies and interaction possibilities. While users can view and explore three-dimensional data stereoscopically, interacting with such data in a three-dimensional space has proven to be difficult. Despite the development of many new input devices in the wake of the second wave of virtual reality [3], no standard has yet emerged for interacting with such objects in a mixed reality environment. Typical approaches which use controllers and gestures hold numerous drawbacks. Problems such as the gorilla-arm-effect and the lack of tactile feedback make good interactive handling in 3D space difficult.

This master thesis investigates the use of three-dimensional interactions using a touch-sensitive tablet, in order to determine if such are suitable for exploring a three-dimensional object in AR. The focus lies on the interaction possibilities, the usage of a tablet would offer, which are intended to make it easier for the user to deal with volumetric data in AR through spatial and touch-based input. This hybrid form of input offers an intuitive way to visualise modified volumetric data in the user's real environment and to manipulate and explore it in a meaningful way through intuitively designed actions.

First, the foundation for the topic is laid by presenting related projects which also deal with interacting with 3D data. Then, different input devices are described before the interaction techniques themselves are discussed. The gathered information is then used to design a concept for an AR-based prototype using interactions based on touch-based and spatial gestures, while the panel-like shape of the tablet is used for intuitive inspection features. The design of this application is being incorporated into a corresponding prototype, which is evaluated in the course of a qualitative user study.

The focus of this study lies on how the user perceives the usage of the tablet as an input device and the possible actions which can be applied to virtual objects. The result showed that a tablet is predominantly perceived as a good and intuitive input device. In addition, the handling of touch-based and spatial gestures was understandable. When examining the data set, two-dimensional data could be stored from the three-dimensional model. All study participants agreed that the use of a marker to indicate the origin of the data was sufficient to draw a connection between 2D and 3D and to understand the information context of the two-dimensional data.

Kurzfassung

Seit Augmented Reality (AR) in den 1990er Jahren das erste Mal erwähnt wurde [4], hat sich die Domäne im Bezug auf vorhandene Technologien und Interaktionsmöglichkeiten stark weiterentwickelt. Obwohl es Nutzern erlaubt drei-dimensionelle Daten stereoskopisch zu betrachten und zu erkunden, erweist sich die Interaktion mit solchen Daten im drei-dimensionellen Raum als schwierig. Trotz der Entwicklung vieler neuer Eingabegeräte im Zuge der zweiten Welle der virtuellen Realität [3], hat sich noch kein Standard für die Interaktion mit solchen Objekten in einer Mixed-Reality Umgebung hervorgetan. Typische Ansätze, welche Controller und Gesten verwenden, bergen zahlreiche Nachteile. Probleme, wie der Gorilla-Arm-Effekt und fehlendes taktiles Feedback erschweren eine gute interaktive Handhabung im 3D-Raum.

In dieser Masterarbeit wird der Einsatz von drei-dimensionalen Interaktionen mit einem berührungsempfindlichen Tablet untersucht, um festzustellen, ob diese für die Erkundung eines drei-dimensionalen Objekts in AR geeignet sind. Die Nutzung solcher Interaktionsmöglichkeiten soll Nutzern den Umgang mit volumetrischen Daten in AR durch räumliche und berührungsbasierte Eingaben erleichtern. Anwendern bietet sich mit dieser hybriden Eingabeform eine intuitive Möglichkeit, um modifizierte volumetrische Daten in der realen Umgebung der Benutzer zu visualisieren und diese durch intuitiv gestaltete Aktionen zu manipulieren und sinnerfassend zu erkunden.

Zunächst wird die Grundlage für das Thema gelegt, indem verwandte Projekte vorgestellt werden. Anschließend werden unterschiedliche Eingabegeräte beschrieben, bevor auf die Interaktionstechniken selbst eingegangen wird. Die gesammelten Informationen werden für den Entwurf eines Konzepts für eine AR-basierten Prototypen verwendet, welcher Interaktionen beruhend auf touch-basierten und räumlichen Gesten konzipiert und die panelartige Form des Tablet für intuitive Inspektionsarten nutzt. Der Entwurf dieser Applikation wird in einen entsprechenden Prototypen eingearbeitet, welcher im Zuge einer qualitativen Nutzerstudie evaluiert wird.

Hierbei liegt das Augenmerk vor allem darauf, wie Anwender die Verwendung des Tablets als Eingabegerät und die möglichen Aktionen, welche auf virtuelle Objekte angewandt werden können, empfinden. Diese Studie zeigte, dass ein Tablet überwiegend als gutes und intuitives Eingabegerät wahrgenommen wird. Zudem war die Verwendung von touch-basierten und räumlichen Gesten verständlich. Bei der Untersuchung des Datensatzes konnten zwei-dimensionale Daten aus dem drei-dimensionalen Model gespeichert werden. Alle Testpersonen stimmten zu, dass die Verwendung eines Markers für die Kennzeichnung des Datenursprungs ausreichend ist, um eine Verbindung zwischen 2D und 3D ziehen zu können und somit den Informationskontexts der zwei-dimensionalen Daten zu begreifen.

Chapter 1

Introduction

The traditional approach to visualise, examine, and compare two- and three-dimensional data is to use a desktop computer system or a tablet. Such systems are characterised by their monoscopic rendering capability, a limited *field of view* (FOV) and a lack of head tracking. Furthermore, the presentation of data is restricted to the display size and offers only a limited layout space. Even though it is a conventional hardware, it is hard to use for manipulating, exploring, and understanding three-dimensional data.

Over time, technologies from the field of *Mixed Reality* (MR) [60], such as *Virtual Reality* (VR) [77] or *Augmented Reality* (AR) [4], have made it possible to get a new perspective on the visualisation of three-dimensional data. In combination with a *Head Mounted Display* (HMD), AR allows an application to add virtual aspects, or even full data sets, to the user's real environment, which can then be perceived stereoscopically and interacted with in real time.

Such three-dimensional data can be obtained by using an image generation method such as *Computer Tomography* (CT) or *Magnetic Resonance Imaging* (MRI) [82]. The result can be expressed as volumetric data, which is a set of samples representing specific values in three-dimensional locations [43]. As this kind of data contains a lot of information, it is very extensive and therefore difficult to process [51]. Since the processing of such data sets requires a lot of computing power, it is hard to display them in an augmented environment, as the response time of the program is slowed down, which undermines the real-time responsiveness of an AR program.

The handling of such data in AR is difficult. According to Mine [61], there are three ways a user can interact with three-dimensional data: direct user interaction, physical controls, and virtual controls. While direct user interactions, such as mid-air gestures, may lead to fatigue [36] and confusion [24], virtual controls, may allow great flexibility but lack haptic feedback and may complicate interaction [61]. Physical devices, such as *Hand Held Devices* (HHD), on the other hand, may enhance the feeling of presence while often lacking a natural mapping.

The choice of interaction possibility in AR is tough if two-dimensional data is to be displayed in addition to three-dimensional data. This combination should allow the user to explore monoscopic data in the spatial context of the stereoscopically rendered data and get a better insight into the information presented. However, it is challenging to create an intuitive and simple way for the user to deal with both formats.

1.1 Research Questions

Based on this starting point, the following question has emerged:

"How to intuitively interact with and explore three-dimensional data in AR using a spatially aware, touch-sensitive HHD."

Since one can handle both, two- and three-dimensional data, when working with a combination of AR and panel shaped HHD, the following meta question has also arisen:

"How to query and compare 2D data from 3D data while maintaining the connection to the 3D context."

In order to answer these questions, it is first necessary to examine which kind of interaction possibilities are available and which projects have already been carried out which deal with the investigation of 3D objects in a *Mixed Reality Environment* (MRE). With the knowledge of the different existing techniques, a concept can be developed and implemented in a prototypical implementation. A qualitative evaluation of this application allows to detect which aspects of the concept are intuitive and if the chosen approach was appropriate for the interaction between user and volumetric dataset.

1.2 Overview

Chapter 2 The *Related Work* chapter deals with the investigation of similar projects and different input and output possibilities in an MRE. In addition, different interaction techniques and tasks are explored to form a good basis for the following concept chapter.

Chapter 3 The *Concept* chapter builds on the findings of chapter 2. With the help of the comparisons and various design guidelines, a concept for an AR program is designed which allows the user to interact with and explore volumetric data.

Chapter 4 The *Implementation* chapter describes how the previously established concept is realised in the form of a prototypical application. The technologies used are introduced before the data and implementation details are explained.

Chapter 5 The *Evaluation* chapter describes how the prototype is tested and evaluated in the context of a preliminary qualitative user study. The findings are discussed and are intended to provide a first insight into the answers to the research questions.

Chapter 6 The final *Conclusion* chapter summarises the previously drawn conclusions on the focus of the master thesis before it closes with a list of possible extensions.

Appendix A The first appendix holds the list of acronyms used in this master thesis.

Appendix B The second appendix is a collection of all documents used for the evaluation in chapter 5. It contains the information about data privacy, the user questionnaire, as well as the interview lead questions for the concluding conversation with the study participants.

Chapter 2

Related Work

The first section 2.1 of this research summary covers the topic of closely related projects, which use a touch-sensitive *handheld device* (HHD) in a *mixed reality environment* (MRE) to manipulate a three-dimensional data set. Opposed to a conventional desktop interaction, there is a broad variety of possibilities to interact with volumetric data in 3D space. The choice of input device has a great influence on the design of the interaction technique, so section 2.2 describes different input devices which can be used to handle 3D data sets. The following section 2.3 introduces the concept of virtual displays which allow virtual output. Section 2.4 then discusses different types of interaction techniques before closing the chapter with section 2.5, which covers interaction tasks.

2.1 Closely Related Work

This section focuses on projects which concentrate on the utilisation of HHDs when manipulating and exploring three-dimensional data in *Augmented Reality* (AR) or *Virtual Reality* (VR). In the first subsection 2.1.1, publications with a focus on the utilisation of HHD without computing power are covered, before addressing those, which take advantage of devices with computing powers, more specifically touch-sensitive tablets (see subsection 2.1.2).

2.1.1 Physical Props as Handheld Devices

A pioneer project investigating the interaction of volumetric data in AR has been the *Studierstube* project. This project utilises the *Personal Interaction Panel* (PIP) [80], a combination of tracked panel and pen, to interact with three-dimensional graphics. Multiple publications have been published developing and extending the Studierstube [71, 72, 81, 86]. While the original Studierstube publication was only the introduction of the idea for a multi-user AR environment [72], the idea was picked up and developed further when Szalavari et al. introduced the PIP the following year [80]. They planned interactions allowing the user to use the panel as a cutting plane or for holding tools for further manipulations. In 1998, Szalavari et al. used see-through *Head Mounted Displays* (HMDs) to present three-dimensional stereoscopic graphics to multiple users simultaneously, which allows natural communication and interaction between co-located users [81]. Additionally, the PIP was used for the direct interaction with presented 3D

models (see figure 2.1). Incorporated layers allow the users to toggle the visibility of different layers, which show different sets of aspects. Further, annotations can be added to the model onto the 3D model using the PIP, and can be handled with an intuitive drag and drop mechanisms in 3D. The system works with tracked mobile objects which allow tangible interaction. While the PIP is a tangible manipulation tool, physical models can also be used and passed between the users. The combination of HMD and PIP allows intuitive three-dimensional viewing and manipulation which is superior to the conventional screen-and-mouse based interaction when it comes to complex 3D models [81].

Wohlfahrter et al. have extended the Studierstube project using interactive volume exploration [86]. While the original PIP uses a see-through plastic palette and a pen, which are both tracked, Wohlfarter et al. based their prototype on an extended version suggested by Bimber et al. [10, 11]. A semi-reflective, semi-transparent foil was used on the panel, which allows the system to calculate volume projection by using a focus or eyepoint which has been reflected over the pad. This extension enables the 3D space to be enlarged and also minimises the distortion of the projection for users other than the one holding the PIP. The extended project allows the user to scale the rendered volume using a slidebar, which is mapped onto the PIP, while the volume can also be grabbed and rotated. The clipboard form of the PIP allows it to be used as a cutting plane (see figure 2.2). When moving it through the model, cut parts are removed so the user can see the internal structure of the data set. As the movement can be performed with the non-dominant hand, the pen of the PIP can additionally be used to move the model. Cutting planes can be frozen, hereby a copy of the clipping plane is stored and fixed to the volume. Frozen planes can also be unfrozen to reset the model. Further, the system allows the user to fast and easily extract arbitrary slices of the volume.

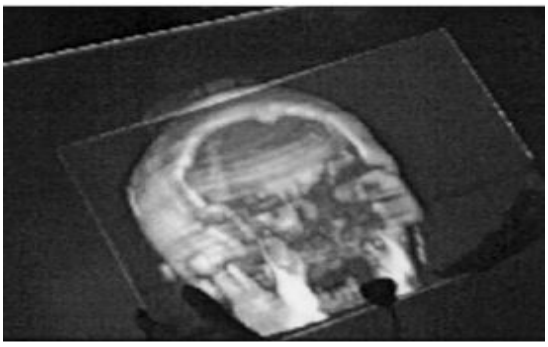


Figure 2.1: Positioning a clipping plane with a tracked panel [71].

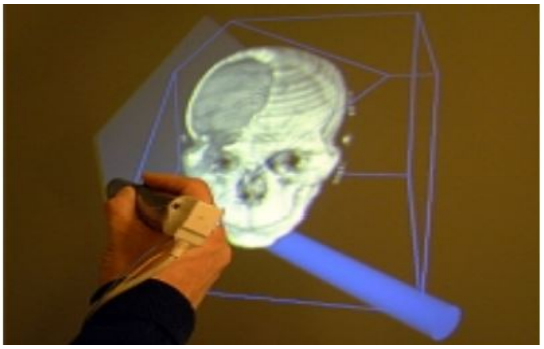


Figure 2.2: Positioning a clipping plane with a tracked pen [86].

The PIP was a popular manipulation tool to be used for interaction with volumetric data. Bornik et al. used it for computer-aided liver surgery planning in 2003 [15]. In combination with a see-through HMD, stereoscopic visualisations could be analysed in AR, using the PIP for input actions. The data consisted of surface or volumetric models where the system automatically segmented *Computer Tomography* (CT) data sets as input data. When moving the tracked pad through the model, it could be used as a clipping plane. The original CT data is shown on the surface, while the model is clipped

above the HHD and thus revealing the internal structure. The position of the panel within the model can be saved in form of a snapshot which could be placed freely in the user's environment (see figure 2.3). Further, the pen of the PIP allows to take distance measures on the model. This project has been extended the following year with more focus on the modification of the volumetric data using the pen [16]. While the panel was still used for immediate inspection, the pen was used to specify regions of interest and to modify the selected segmentation (Vertex-Drag Tool, Free Deformation Tool).

Cassinelli and Ishikawa also introduced a device to interactively explore volumetric data [22]. Other than the PIP, the control interface can be created with a combination of a Plexiglas® or translucent paper, which is equipped with ARToolkit[1] markers. This handheld interface in combination with a passive projection screen allows the slicing of the projected model into saggital, coronal, and horizontal slices as seen in figure 2.4. The markers are used to track the position and orientation of the handheld publication, so the corresponding slice can be calculated. Additional markers enable the user to switch between modes which enable the taking of snapshots, or dragging of the position within the virtual volume. The images are fixed, so the rotation of the publication with respect to the volume model is not possible. The volume slicing display allows a workable space as large as the user's environment without the need of an HMD. Even though the screen can be flexible, only straight slices can be displayed.

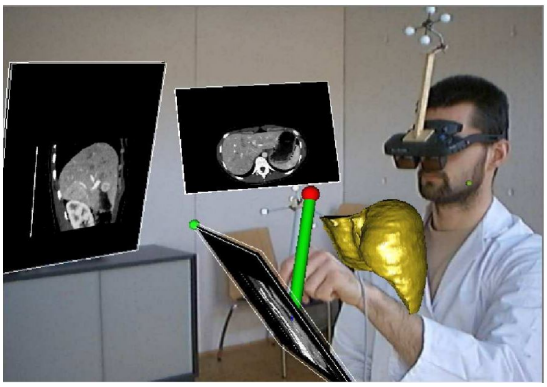


Figure 2.3: Creating snapshots from cutting planes displaying CT data [15].

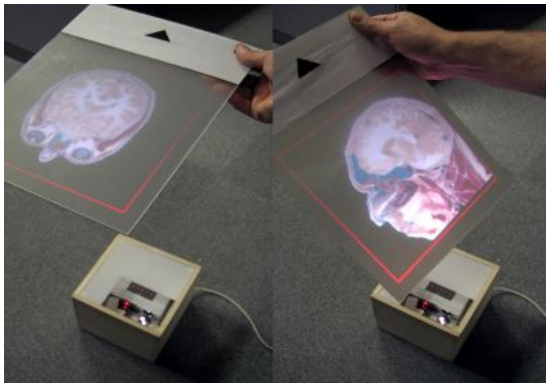


Figure 2.4: Slicing a 3D data set using passive projection on a Plexiglas® [22].

2.1.2 Handheld Devices with Computing Power

Issartel et al. combined a touch-sensitive HHD with tangible tools which allows a direct, natural, and full six degree of freedom (6-DOF) interaction [40]. The tangible objects are a reference object to represent the data set, a stylus which can be used as a multi-purpose exploration device (such as a "virtual blade") and a tablet. The tablet can be used as a see-through window which allows tactile feedback. The data set is superimposed on the reference object, so it rotates and translates along with the movement of the reference object. The volume data can be sliced to visualise the structures within, using the planar

[1] http://www.hitl.washington.edu/artoolkit.html

shape of the tablet to control a tangible virtual cutting plane. The display of the tablet allows to show the progress of slicing real-time through the handheld window.

A combination of multi-touch wall display and a handheld device (iPod Touch[2]) was used by Song et al. to allow the visual exploration of volume data [76]. A slicing plane can be rendered in the virtual object and directly or remotely manipulated. The direct rotation can be executed by rotating the HHD against the wall display while the translation happens by using one finger to translate the slicer on X and Z plane or two fingers to move it in Y direction. The idea is to give the user the feeling of holding a physical slicer in hands. The visualisation can be rescaled using the conventional pinch gesture. Song et al. made use of the 3D-tilt sensing and multi-touch capability on the HHD to allow direct and efficient manipulation of a slicing plane within the visualisation.

Lopez et al. used a touch-sensitive tablet to allow as much engagement as possible when exploring a 3D data set [52]. The idea is to combine a large immersive view of the data sets and control this view with touch input on a touch-sensitive interface (see figure 2.5). The input on the mobile device is combined with the control of Android's virtual rotation sensor. 3D navigation commands need to be initiated via touch and then are controlled over a tablet which syncs the immersive and tablet views. This enables the user to move around in the room and retrieve alternative views of the data which allows an improved interaction experience. The user can explore two data sets through tablet-based interaction.

Sereno et al. studied how collaboration can be enabled by combining an HMD with a multi-touch device [73]. Hand tracking is used to apply ray casting, to indicate 3D locations where the tablets 2D input is used to execute 3D manipulation. The combination enables view manipulations as well as the adding of text, images, and hand-drawn sketches. Co-located collaborative exploration can be seen in figure 2.6.

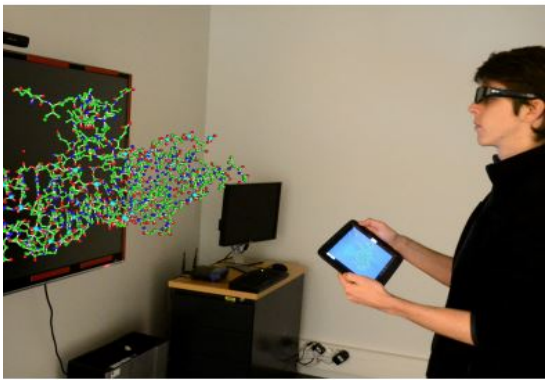


Figure 2.5: Touch input manipulating stereoscopically visualised data [52].

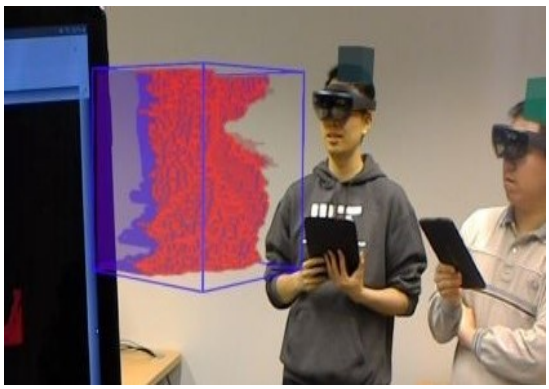


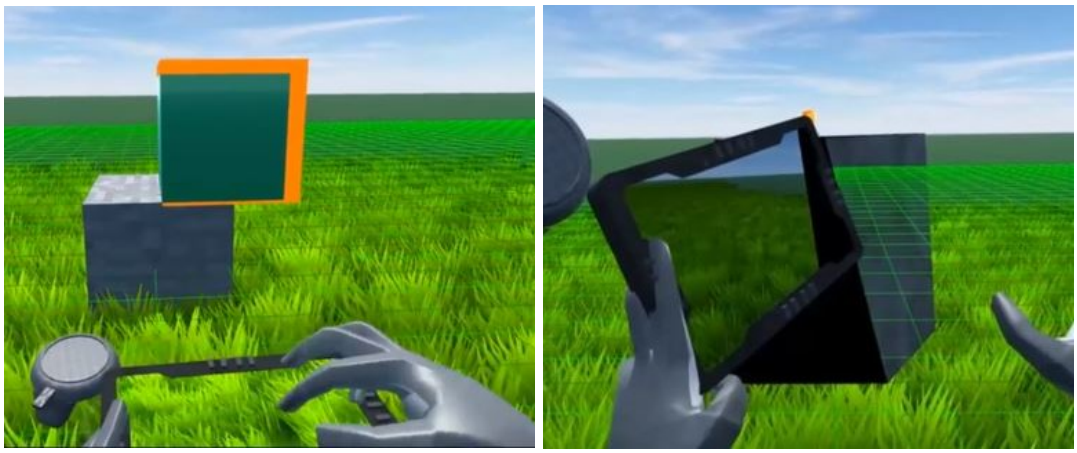
Figure 2.6: Sereno et al. collaboratively manipulate data using touch input [73].

Surale et al. show how exploring a space using a multi-touch tablet in VR can be implemented [78]. In their publication *TabletInVR* a 3D-tracked multi-touch tablet is used in an immersive VR environment to support 3D solid modelling. It is depicted how the touch input capability and physical shape can be utilised, as well as the benefits of

[2]https://www.apple.com/ipod-touch/

metaphorical associations, and the natural compatibility with bare-hand mid-air input. The prototype enabled users to create objects by flipping the tablet with the non-dominant hand and choosing the respective object from a scrolling list. The placement is executed by pointing the ray or using a mid-air pinch. Created objects can be selected by using the tablet's ray, executing a pinch gesture when piercing it with the tablet or tapping on the face of the respective object (mid-air). Multiple objects can be selected with a knuckle hand posture. Actions can be exited and objects deleted, using a "swipe-in" movement. Text annotations are interactive objects which can be selected and re-arranged. A metaphorical association in this prototype is the usage of the tablet as a physical "knife" which can trim parts of objects (see figure 2.7a). Navigation can be initiated with a five-finger touch on the touch interface, using the tablet as a view port.

Simulator and motion sickness is mitigated with a limited *field of view* (FOV). A help feature can be activated via voice recognition which then displays a help video. A user evaluation conducted that the gestures were "very intuitive and easy to follow". Actions such as create, delete, and modify were also intuitive while they sometimes unintentionally created objects. Transformations were found to be easy while rotations, which were executed by rotating a hand on the touchscreen (see figure 2.7b), were challenging as the finger distance could be recognised as scaling. Some tasks also required to flip the tablet with the non-dominant hand which caused discomfort and was perceived to be cumbersome.

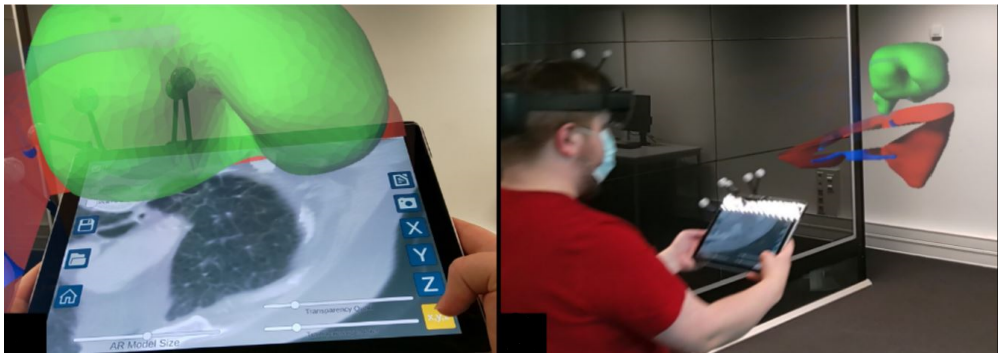


(a) Rotation (b) Slicing ("knife" gesture)

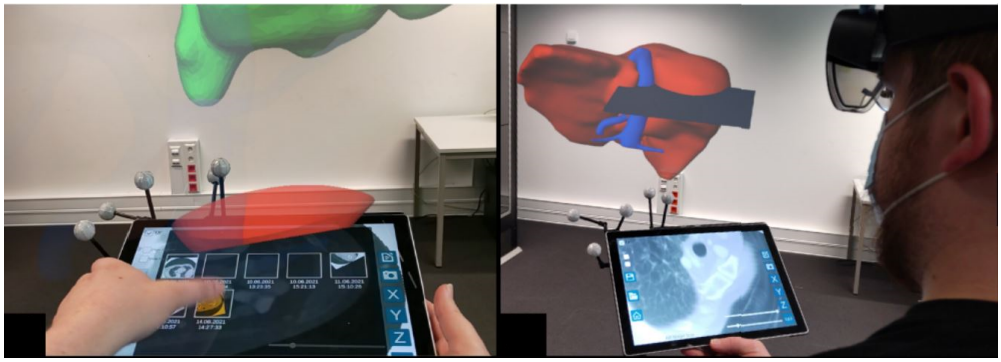
Figure 2.7: TabletInVR allows the user to interact with a virtual object using a touch-sensitive tablet [78]. a) Rotations are executed by performing a twist gesture on the display. b) The selected object can be clipped by handling the tablet as a "knife".

In contrast to Surale et al. [78], Luo et al. made use of a spatially tracked tablet in combination of an HMD to interact with a model in AR [55]. The focus of this publication lies on the spatial interaction with and exploration of volumetric data. They allowed the interaction using three-dimensional input as well as touch input. To set the initial position of the model in the user's environment, they mapped it to the rotation and translation of the tablet. Additionally, they used multiple buttons to execute actions

such as starting a transformation mode, freezing the current cross-section (see figure 2.8a), or locking an axis for further examination. Overall, they offered solutions for spatial manipulation, such as transforming and rescaling of the data set, exploration using cutting planes (see figure 2.8b), as well as measuring actions. The prototype further allows to capture slices, which could be viewed in a gallery mode or be annotated. Luo et al. believe, that the user could benefit from the utilisation of three-dimensional interaction when exploring and slicing volumetric data, in the sense of a better spatial understanding as well as a reduced cognitive load.



(a) Freezing a slice



(b) Revisiting captured slices

Figure 2.8: Exploration of a three-dimensional data set in AR [55].

2.2 Input Devices

Traditionally, desktop computer systems are being used to render and analyse three-dimensional data. This setup is characterised by a monoscopic view and the lack of head-tracked rendering [48]. The use of immersive technologies makes it possible to display spatially complex structures in a way that makes it easier for the user to analyse, understand, and explore data [20]. As already described by Mayer et al. [59], there are multiple input devices which are favourable for the interaction with three-dimensional or volumetric data. Often it depends on the task at hand to find the best suited device.

This section introduces various generic input devices which were designed to interact with 3D data sets in subsection 2.2.1, before describing panels and tablets in detail in subsection 2.2.2.

2.2.1 Generic Input Devices

Over the years multiple devices have been designed and developed to facilitate the interaction with three-dimensional data. This subsection describes some of these devices.

A well-known approach is the *cubic mouse*, which has been developed by Fröhlich and Plate in 2000 [33]. As can be seen in figure 2.9, this cube shaped box is crossed by three perpendicular rods which represent the X-, Y-, and Z-axis respectively. Buttons on the top allow additional control, while a 6-DOF tracking sensor is used to control the position, as well as the orientation of a mapped virtual model. Additionally, each rod moves a slicing plane along the corresponding axis. The simulations were displayed using virtual environments such as a responsive workbench [46, 47], a CAVE [27] or a large screen projection installation. The cubic mouse was tested analysing volumetric data sets from *Magnetic Resonance Imaging* (MRI) or *Computer Tomography* (CT) scans. The results showed the ease of use for users. The disadvantage of the design of the cubic mouse, is the mapping of the device's movements, which is often no longer intuitive as soon as the device is rotated [33].

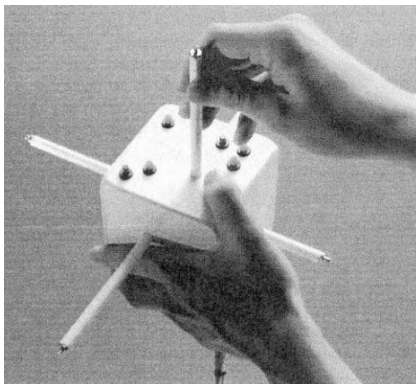


Figure 2.9: Cubic Mouse with tree perpendicular rods which can be used for the translation of cutting planes [33].

In a subsequent publication about projects in VR, Fröhlich et al. analysed multiple input devices and conducted a study to show which are the most efficient in terms of applicability with 3D interfaces [32]. The main requirements for these devices were that they support at least 6-DOF and are operable in three dimensions. It was differentiated between desktop devices such as the *GlobeFish* [31] and the *GlobeMouse* [31], and handheld devices such as the *GlobePointer* [32] and the *SquareBone* [39]. The GlobeFish, combines the rotational characteristics of a 3-DOF trackball with elastic translation (see figure 2.10), while the GlobeMouse combines this 3-DOF trackball with the commercial SpaceMouse sensor which handles the translational input (see figure 2.12) [31]. The

SpaceMouse is an input device developed by 3DConnexion for 3D navigation (6-DOF) in computer-aided design (CAD) programs [3].

The GlobePointer is based on the idea of the GlobeFish, but designed as a handheld device as can be seen in figure 2.11 [32]. The HHD measures the hand orientation using a 2-DOF gyroscopic sensor, while two joysticks (1-DOF and 3-DOF) are used to manipulate the depth translation and the rotation of the selected object. The *two-4-six* device utilises a touchpad, a gyro- and elastic sensor for translations, while the arcball approach [74] is used to measure rotations. The SquareBone has a handheld design in form of two 6-DOF SpaceMouse sensors and a tracking sensor were designed which could be manipulated with the fingertips of both hands (figure 2.13) [39]. An extended user study was conducted comparing these devices and the cubic mouse [33] in tasks requiring navigation as well as object manipulation [32]. While the cubic mouse was preferred by novice users, the SquareBone performed best. It was stated, that specific applications often need specific device requirements and that it might be hard to find a universal 3D input device which works for all applications.

Figure 2.10: The GlobeFish is a 6-DOF controller with a 3D trackball [31].

Figure 2.11: GlobePointer which is based on the GlobeFish [32].

The *roller mouse* was developed in 1993 by Venolia et al. to arrange 3D objects in a scene [83]. It is based on a standard 2D mouse but includes two wheels on the front, as can be seen in figure 2.14. The mouse is used to control the tip of a cone shaped 3D cursor which changes its orientation as it moves while the body gets dragged behind ("tail-dragging"). Tail-dragging enables the translation and rotation of 3D objects (yaw and pitch orientations). An intuitive model of magnetic attraction with a similar snap-top behaviour such as "gravity fields" can be used to help align objects [7, 9]. The roller mouse enables the user to move the cone courser in three dimensions simultaneously.

Another alternative to conventional input devices is Bornik's *Eye of Ra* which can be used in combination with a tablet PC [14]. It holds a stylus tip, which allows direct 2D interaction on the tablet, while additional buttons and a scroll wheel are used for 3D interaction (see figure 2.15). The Eye of Ra allows the user to make two-dimensional manipulations with a focus on high precision, and three-dimensional manipulations which

[3] https://3dconnexion.com/de/spacemouse/

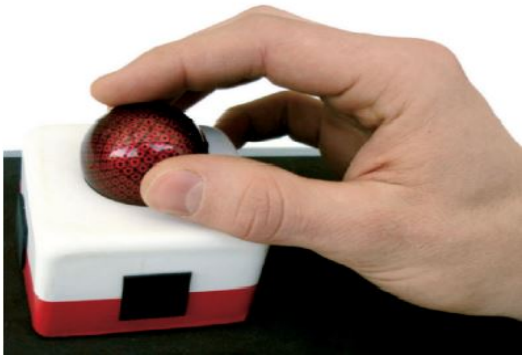


Figure 2.12: The GlobeMouse is a tracked 3D ball incorporated on a 3D translation controller [31].

Figure 2.13: The SquareBone is a two-handed device, which consists of two 6-DOF SpaceMouse devices [39].

allow high speed input. The device can be used with a power or a precision grasp, which allow the user to concentrate on different tasks. It was specifically developed as a template shaping tool for deformations, segmentation, labelling data, and taking measurements. Tracking targets on the device allow 6-DOF tracking.

In contrast to the previous publications, Gallo et al. took advantage of the already existing *Nintendo Wii* remote controller [4] [34]. This HHD was used as a three-dimensional user interface for manipulation and pointing tasks when interacting with volumetric medical data in a semi-immersive virtual environment.

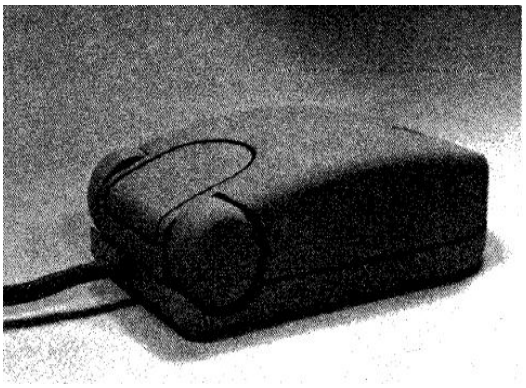


Figure 2.14: The roller mouse can be used in three dimensions [83].

Figure 2.15: Eye of Ra can switch between 2D and 3D interaction [14].

[4] https://www.nintendo.co.uk/Wii/Accessories/Accessories-Wii-Nintendo-UK-626430.html

Device name	Input	DOF	Citation
Cubic mouse	Button input, translational input from three perpendicular rods, position and orientation input via tracking sensors	6-DOF	[33]
Globe Fish	Elastic translational input and rotational input via a 3-DOF trackball	6-DOF	[31]
Globe Pointer	2-DOF gyroscopic sensor for translation input, 1-DOF joystick for depth translation, and a 3-DOF joystick for rotational input	6-DOF	[32]
SquareBone	2 rotational and 2 translational input sensors	12-DOF	[39]
Globe Mouse	3-DOF rotational input via a trackball and translational input via an elastical 3D translation controller	6-DOF	[31]
two-4-six	The touch input on a touchpad is used for rotations, while a gyrosensor in combination with an elastic sensor are used for translations	6-DOF	[32]
Roller mouse	Mouse ball encoder for transitional input and two wheel inputs	4-DOF	[83]
Eye of Ra	Buttons input, spatial tracking, and input via an integrated stylus	6-DOF	[14]
Nintendo Wii remote controller	Position and orientation input via infrared tracking	6-DOF	[34]

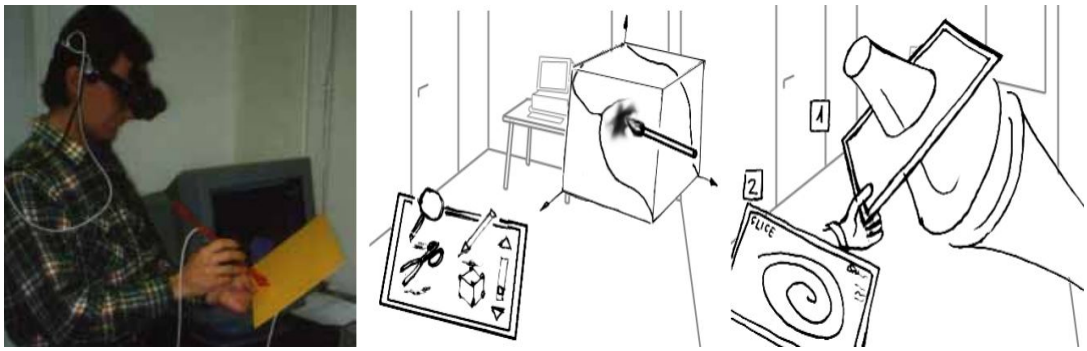
Table 2.1: Generic input devices overview.

2.2.2 Panels and Tablets

In contrast to the input devices previously presented, this subsection introduces panel shaped handheld units, which can be used to display data, and how they can be utilised in a three-dimensional application.

Angus et al. made use of a handheld flat panel or physical clipboard in combination with a stylus, which were both registered with position trackers in the virtual environment [2]. The *Personal Digital Assistant* (PDA) allows a 3D interaction metaphor with 2D user interface tool kits. Tools aligned to the panel are within easy reach and do not clutter the users FOV, while the panel allows the system to embed 2D interactions into the users 3D environment. The panel can be used to display menu options, with the menu attached to the panel and not floating in the environment like other VR menus. The advantages of this set up are the high fidelity, as well as the natural feedback and the option to put the prop away to clean up the FOV as the virtual elements are attached to the clipboard. Although the panel allows to embed virtual elements in the users physical environment, it also restricts the size of the display to the physical borders of the handheld panel.

Another two-handed interface was introduced by Szalavari et al. [80], the previously mentioned *Personal Interaction Panel* (PIP). This panel allows easy-to-understand manipulation tasks in augmented and virtual environments. Due to the constraints of the technology at the time of publication and demanding hardware such as LCD shutter glasses and pressure-sensitive displays, it was decided to set up the PIP using simple hardware. This system can be implemented using a plain panel and pen along with a see-through HMD. Tracking markers are placed on pen and pad as seen in figure 2.16a, as the hardware itself does not hold any computing power or sensors. The big advantage of this setup is the passive haptic tactile feedback which enriches the interaction. It is designed for the user to hold the panel in the non-dominant hand while the dominant hand can use the pen for basic object manipulation tasks such as selection, transformation, zooming, layering, or browsing remote environments. Further, it enables the user to use the panel as a tangible cutting plane, slicing the model (see figure 2.16c), and creating snapshots. Further interaction possibilities can be chosen in a menu displayed on the panel (see figure 2.16b). Users who tested the prototype did not report any problems with fatigue and found the user interface natural to interact with.



(a) Setup (b) Tool menu (c) Slicing

Figure 2.16: Personal Interaction Panel (PIP) introduced by Szalavari et al. in 1997 [80] a) The PIP consists of a tracked panel and pen which are augmented using an HMD. b) The PIP can be used to hold interaction tools, c) or to slice virtual objects and display internal structures.

The next step from so called "dumb panels", which do not hold any computation power such as the PIP, to the modern touch-sensitive tablet, was the utilisation of handheld monitors. Fitzmaurice first made use of spatially aware palmtop computers to utilise physical objects as anchors for information when creating a prototype named *Chameleon* [29]. Such handheld monitors situated information in a physical context in form of an information space, which could hold multimedia (text, video, graphics, audio) [30]. The additional information about the organisation of space could also improve the users orientation. Wellner then introduced the term *computer-augmented environment*, which describes the merging of electronic systems into the physical world instead of replacing them [85].

Amselem and Rekimoto researched and extended Fitzmaurice's idea of a handheld display, which is aware of its position [30] for the interaction with the virtual world [1,

68]. Amselem captured the position and orientation of an HHD using a 6-DOF tracking device and mapped it one-to-one to the viewpoint in space [1]. This tracking enabled the encoding of "real" and "information" space which helps to avoid an overflow of information and associate electronic information with physical objects. In 1995, Rekimoto and Nagao created the *NaviCam*, a device which could detect colour code IDs placed in the environment and display related information [68]. It combines ID-awareness with a portable video-see-through display, which uses information from the user's environment as implicit input. Such *Graphic User Interfaces* (GUI) on HHD allow to reduce the cognitive overload even though they do not reduce the volume of operation itself.

Workbenches used to be a suitable alternative to HMDs, as they had a higher resolution and were a cost-effective alternative to CAVEs which are very expensive. Therefore and due to the ability to display data in real-world scale, it is suitable for medical VR applications [51]. Schmalstieg et al. combined a workbench-like setting with transparent props in form of a tracked hand-held pen and a pad [70], which are related to the PIP [80]. The pad is usable as a palette which can carry controls and tools while also functioning as a window-like see-through surface. Interactions took the form of active object manipulations or the creation of snapshots of the real 3D scene. The pad works as a "window" which can display the 3D graphics and therefore represents an embedding of 2D in 3D, while allowing the manipulation of seen objects. The selection of such objects is solved with a fish net selection, which works by intersecting desired objects with the pad. Replicas of selected objects appear on the panel, ready to be changed.

Device name	Display	Input	Citation
PDA	Tracked panel, no computing power	Virtual finger or stylus, tracked	[2]
PIP	Tracked panel, no computing power	Tracked pen	[80]
Chameleon	Palmtop computer	Gestures and one button	[29]
Hand-held display	LCD display	Orientation and position with a 6-DOF tracker	[1]
NaviCam	Portable computer	Video stream, gestures, and one button	[68]
Transparent props	Tracked panel, no computing power	Tracked pen	[70]

Table 2.2: Panels and tablets overview.

2.3 Virtual Displays

While physical devices are restricted to their size [2], virtual displays allow to extend their functionality of presenting information to the user.

Virtual displays are windows which are rendered in AR or VR to show data. They can be stand alone or enlarge an existing screen. Normand and McGuffin explored the idea to enlarge the screen of a mobile device with the help of AR, thus creating a *Virtually Extended Screen-Aligned Display* (VESAD) [62]. They aligned virtual surfaces

co-planar with a smartphone and used the handheld device for interaction options (see figure 2.17). The windows are mapped to the device so they stay aligned when the device is moved. Interaction is possible through a quick rotation of the device that switched information, as well as through "slide-and-hand", where the information displayed on the screen is detached and displayed mid-air with AR. It was compared how the users fare when using the phone as input but VESAD as output, the phone and VESAD as output but mid-air gestures as input, and only the phone. This comparison showed that using AR to expand the output window, but with the ability to use the mobile device for input, was significantly faster and preferred over the other two techniques.

Another publication focusing on embedded AR visualisations was contributed by Reipschläger et al. [66]. They proposed to display data in AR as an extension of a large interactive display to allow further exploration and a different view on the user's design space. Similar to the VESAD, these embedded AR visualisations can extend directly from the physical display. To avoid difficulties in reading the original data due to overloaded and complex additional information displayed in AR, they proposed to limit and sensibly select the external data.

Figure 2.17: VESAD maps virtual windows to the user's mobile device [62].

In 2020, Biener et al. investigated in a publication how to extend 2D displays into the third dimension [6]. They investigated the combination of an HMD with touch-sensitive laptop or tablet to explore the joint interaction space in VR. While virtual windows are often spatially arranged in scaffolding geometry (e.g. spheres or cylinders), they suggest a depth layering to allow the usage of this technique in a constrained environment. Is is suggested to use the touchscreen of the tablet for interacting with and switching between the virtual windows. An alternative could be a combination of gaze tracking and touch gesture using a temporal threshold to avoid unwanted interactions (Midas touch problem [41]). An executed design parameter evaluation found that the gaze and touch combination outperforms the bi-manual method by approximately 30% in terms of speed, while also achieving higher ratings on the *System Usability Scale* (SUS [18]). The interaction with a selected screen should either be executed by using the touchscreen as indirect input device or by aligning the virtual display with the physical touchscreen. The switching between screens could be achieved by using a two-finger swipe gesture [6].

The behaviour of users when interacting with virtual displays using a smartphone was investigated by Ren et al. in the context of a formative interview [69]. Many possibilities on how to manipulate virtual windows in AR using a touch-sensitive 6-DOF device have emerged. Ren et al. found that users consider view management using a smartphone as more efficient and accurate compared to hand gestures. Touch, posture, position, and movement were used as input methods on the touch interface. A pull gesture with two to three fingers or the side of a palm was used to create a windows aligned to the smartphone, while a throwing gesture was utilised to create a view next to the user. Further examples for envisioned smartphone postures can be seen in the following figure 2.18. They stated how it was important for the user to "avoid conflicts with the original function" [69].

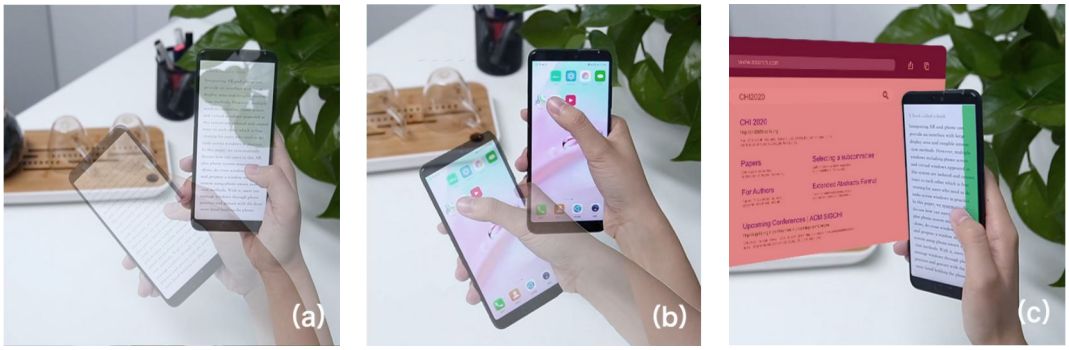


Figure 2.18: The formative interview conducted by Ren et al. concluded following examples [69] a) The smartphone can be shaken to open a window. b) An application can be opened in AR when pressing the app icon while executing a shake gesture. c) If the user holds the edge of the smartphone when it collides with the AR window, the window is to be closed.

2.4 Interaction Techniques

Bowman defines interaction techniques as methods which allow the user to perform a task using a *User Interface* (UI) [17]. Such techniques vary from haptic actions, such as the click of a mouse button, to less tangible actions, such as mid-air gestures. The UI is influenced by the type of input device used. This section first introduces the common interaction techniques mid-air gestures (subsection 2.4.1) and touch input (subsection 2.4.2), before summarising best practices and design insights in subsection 2.4.3.

2.4.1 Mid-Air Gestures

Mid-air interaction is a fast-changing area with many dimensions, where the user can make use of gestures, postures, and movements to interact with a virtual data set [45]. This touchless interaction technique was first mentioned by Sutherland, when describing *the ultimate display* [79]. Executed was the idea by Bolt in the *Put-that-there* project, which enabled the user to move objects with a combination of voice command and pointing gestures [12]. While 3D mid-air interaction allows intuitive, three-dimensional

interaction, gesture interpretation is unreliable and leads to low performances [23]. Additionally, studies report interacting mid-air causes fatigue, also known as *gorilla-arm-effect* [13, 36, 57], while the lack of tactile feedback leads to a greater number of errors and sometimes confusion [24].

2.4.2 Touch and Spatial Input

The concept of touch input was introduced in the context of the first touch-sensitive display by Johnson [42]. The touch wires integrated in the device allow the surface to be sensitive to the touch of a finger. As a touch-sensitive device often offers a hard surface, it gives users a tactile feedback. Spatial input, on the other hand, uses 3D input technologies in free space [37].

Zambramski compared the touch input to the use of conventional tools such as a mouse and pen in a line tracing task [90]. This comparison shows that touch performs better in terms of speed, but does not provide precise contour tracking and is therefore error-prone in comparison as far as replication tasks are concerned.

Lee et al. investigated the usage of a smartphone for the interaction in an MRE [49]. The multi-touch input of a smartphone offers a precise, tactile, immersive, and user-friendly option to manipulate three-dimensional objects. A preliminary user study revealed that this alternative touch input outperforms the conventional interface which combines mid-air gestures and gaze in completion time for a multiple manipulation tasks, such as the placement and transformation of objects, and camera control.

In a 2013 publication, Bruder et al. compared the usage of 2D touch gestures on a touch-sensitive tabletop surface to 3D mid-air interaction [19]. They concluded that touch gestures are most efficient when the object being interacted with is close to the surface. In case of the object being further away than 10 cm from the tabletop display, 3D selection outperforms 2D touch gestures.

When comparing touch input and spatial interaction with 3D data visualisation on a mobile device, Büschel et al. found that three-dimensional interactions alone are more efficient than touch input alone [21]. This comparison of multi-touch and movement had a focus on navigation, comparison, understanding, and memorisation. The spatial interaction was conducted with a tracked tablet and it was found that this method was preferred over touch input as it is more intuitive and dynamic when positioning an object.

In a further study which also compared multi-touch input with device movement, Marzo et al. found that a combination of multi-touch and spatial input works best when positioning and rotating a virtual object [58]. They looked into the manipulation of an object using touch input, device positioning, and a combination of both. Their evaluation showed, that a mixture of movement and multi-touch is superior. This hybrid of spatial and touch input achieves the lowest task completion time and reduces the amount of rotations needed when manipulating a data set. Further, it was found, that small rotations of the virtual object are best done using movement, while touch input is more efficient for large rotations.

Wurm et al. showed how the usage of tablets in a VR application can increase the interaction possibilities as it provides useful input and output options [87]. It enables high-precision selection, clickable interactions, and also provides a menu interface. The

precision provided by the tablet is further more precise than the commonly used wand interface.

Cohe et al. conducted a user study with users with little to no experience in the investigated domain to analyse how they interact with 3D objects using touch gestures [26]. They found that rotation gestures tend to be curved (72,54%), while translation gestures are mostly straight (98,92%). The number of fingers used is not considered to be important, and sometimes more than the minimal number of fingers are used. The experiment showed that users are more confident when they interact with a 3D object using a touchscreen over the use of a mouse. This may be based on the fact that users feel more free to apply strategies using touch gestures over grab gestures which are typical for mouse-based interactions.

In a publication comparing object selection in combination with interaction techniques scrolling, tilting, and moving, Boring et al. describe the respective weaknesses of these techniques [13]. A mobile phone was used to continuously control a pointer on a remote display by means of these techniques. *Scrolling*, was implemented by pushing the phone's joystick or arrow keys into the respective surface which directs the cursor at a constant speed which allows a high selection time for distant targets. *Tilting*, was realised by mapping the acceleration of the cursor to the tilt of the device, while *moving* allowed the direct mapping between phone movement and pointer movement. The latter two allow a faster selection time as there is no constant speed set for the movement, which also leads to higher error rates. These error rates can be reduced by introducing a "snapping" behaviour. In addition, tilting and moving may also lead to fatigue (gorilla-arm-effect), e.g. in case the user moves the whole arm when using the move technique. Despite its limited movement options (linear and diagonal), the scroll technique was found to be the easiest to use.

2.4.3 Best Practices

Considering the previous sections, a combination of touch input and physical movements are a reasonable choice for handling three-dimensional objects at close range. Tablets are best suited for such tasks, as they offer useful input and output options in addition to touch and spatial input [49, 58, 87]. The following points are summarised best practices which can be drawn from the previous subsection 2.4.2:

- When working with interfaces in a VR application, touch interfaces should be used, as they offer easy to use menus and higher precision input compared to traditional VR interfaces [87].
- For placement and transformation tasks, touch input should be preferred to mid-air gestures, as they are perceived as more precise, tactile, and user-friendly [49].
- When working with touch-sensitive tabletops, touch input is most effective and should be used to interact with virtual objects. If the object is further from the screen than 10 cm, mid-air gestures are preferable for object selection [19].
- For navigation and comparison tasks, spatial input is preferable to only using touch input, as it is more supportive and comfortable [21].
- For object manipulation, a combination of spatial input and touch input should be used. A hybrid input system allows a lower task completion time and helps to reduce device movements compared to spatial input or touch input alone [58].

- When controlling an object on a remote display with an HHD, it is best to map the tilt gesture or general movement of the unit. This allows for quick movement, although a latching behaviour should be used to avoid high error rates [13].
- The trajectory for rotational gestures should be designed curved, while translations should be straight. A conducted user study showed that this is the most common behaviour of users envisioning such gestures [26].
- The number of fingers used for touch input should not be fixed. The user should be allowed to use more fingers than required. [26].

In addition to these summarised best practices, the following design suggestions have been proposed in other publications.

Following their previous publication regarding the usage of a touch-sensitive tablet to explore a 3D data set (see subsection 2.1.2), Lopez et al. set up a design space with multiple guidelines to improve the user's engagement when using a touch-sensitive HHD and a combination of monoscopic and stereoscopic displayed data [53].

- A 7-DOF navigation should be supported, which includes 3D translation, 3D rotation, and uniform scaling.
- A consistent combination of data exploration techniques should be enabled by navigation interactions.
- Information should be displayed actively on an HHD which also allows interaction input. Otherwise, the device could be used for input only, so that it can be used without eye contact and thus no displayed information is needed.
- The user should be allowed to physically move within the stereoscopic environment to allow different perspectives.
- The tracking should be held simple.
- Touch input should be precise and well-controlled.
- Physically holding a touch-sensitive mobile device should be minimised as it leads to fatigue.

In an observational study with expert users Lopez et al. tested a prototype design with regard for these guidelines. No participants reported problems or difficulties when switching between the stereoscopic display using an HMD and a monoscopic display in form of a tablet [53].

Further design insights have been defined by Hubenschmid et al. in the course of a publication, analysing the use of spatially-aware tablets in an AR environment for multi-modal interaction with 3D visualisations [38]. The usage of an HMD allows the user an increased stereoscopic perception along with egocentric navigation, which allows a better understanding of abstract 3D visualisations (see figure 2.19). They argued, that mid-air gestures are tiring, unreliable, and inaccurate, while touch interactions are less exhausting and allow a familiar two-dimensional input. This hybrid user interface, consisting of a *Mixed Reality* (MR) HMD in combination with a traditional input device, such as a tablet, allows interaction without visual contact with the input menu (eyes-free interaction). This enables the user to observe the immediate effect of executed actions in the visualisation. When interacting with a data set, the selected part is highlighted while the tablet has a matching background colour scheme. The selection can be changed with the user's head-gaze, using a threshold of three seconds to mitigate the Midas touch

problem [41]. The eyes-free interaction is implemented by dividing the tablet menu into four large areas which can be distinguished without eye contact, while taken actions are mirrored in a *head-up display* (HUD). When the HHD is held vertically, the tablet lens mode is activated. This allows interaction with a 2D visualisation view of the targeted AR object, while the rotation of the object can be synchronised with the rotation of the tablet. Subtle proxemic interactions are implemented, which allow icons and texts to rotate towards the user while small text is only shown when the user is close enough to read it.

Figure 2.19: STREAM allows the user to use a tablet for a fluid combination of 2D and 3D data [38].

The user study conducted revealed that the tablet was uncomfortable to hold in certain positions, e.g. when rotated 90°, although all participants used this spatial trigger to switch to the tablet view. When tangibly manipulating objects, the participants feared to drop the tablet. The results found in this evaluation lead Hubenschmid et al. to define the following design insights [38]:

- Use spatial triggers as input.
- Avoid too much unnecessary eye contact with the screen of the HHD (eyes-free manner).
- Use AR HUD to show available voice commands.
- Employ a loading indicator in AR via head-gaze when selecting an object.
- When using head-gaze, the user should be provided with an action to skip dwell time.
- 3D object alignment should be supported in MR environments.

A formative study by Surale et al. investigates how users envision the usage of a touch-sensitive tablet in a VR environment [78]. The study found, that the creation of an object is expected to happen by drawing on a tablet and extruding or pushing a menu button, while the selection of an object should be executed with a grab or tap on the tablet. Transformations and rotations would be expected to happen when rotating and moving the hand while a pinch to zoom seems most intuitive for rescaling objects. The modification by slicing is expected to happen with the use of menu buttons while a hand or tablet can execute the slicing motion through the virtual object in the

user's environment. These findings have been incorporated in their prototype, which is described in subsection 2.1.2.

2.5 Interaction Tasks

Bowman separated common tasks in VE applications into four categories: travel, selection, manipulation, and system control [17]. As the idea of this project is to focus on selection (subsection 2.5.1) and manipulation (subsection 2.5.2), these are the categories covered in this section.

2.5.1 Selection

Selection is the task to specify one or more virtual objects for any purpose [17]. Depending on the selection technique, the user may face different challenges. For some techniques this could be the limited reach of the user, for others the choice of small or distant objects [17, 50]. Additionally, it can be tricky to select objects in a virtual environment if the environment is cluttered or the object is occluded [63, 88]. With direct selection techniques, the position of the object must be known, which is not the case with indirect techniques, where the user can select from a list of objects for which the name of the object must be known [17, 61]. This subsection introduces different selection techniques and describes their advantages and drawbacks.

Besançon et al. investigated spatial selection techniques for three-dimensional data sets using touch input and introduced an extended taxonomy focusing on user control [5]. Among other things, the selection metaphor, the target selection type, selection shape creation and selection tool control were compared. The target selection type was previously differentiated in object selection and *region of interest* (ROI) based selection, where Besançon et al. now differentiated between single-object and multiple-object selection. They highlighted that "different tasks and environments call for different levels of control over the shape creation". While ray casting is a good technique for context-dependent selection, it would not perform well for particle data selection.

The fundamental idea of ray casting has been first introduced in the *Put-That-There* publication of Richard A. Bolt, where speech recognition was combined with position sensing, to interact with a graphic interface [12]. The system worked with magnetic field measurements using a pair of spatial sensing cubes attached to the user's pointing arm. This allowed the user to refer to any object or location with a pointing gesture after initiating the action with signal words such as "that" and "there". In 1993 ray casting was again picked up in the form of a ray-firing selection mechanism by Liang et al. [50]. The beam has been attached to the X-axis of a bat-shaped virtual object that can be moved by the user to make a selection. This works well in most situations, but when aiming is difficult due to the size of the object or a long distance between pointer and target. Based on this implementation, Hinckley et al. suggested the ray casting metaphor which allows the user to perform a selection by casting a ray or cone [37]. Mine also refers to ray casting when describing at-a-distance selection, though referring to is as laser beams or spotlights [61]. At-a-distance selection is used when the object is outside of the immediate reach of the user. Mine also marks the selection by the intersection of the ray with a virtual object (see figure 2.20).

Extending the ray casting form described before [37, 50], Pierce et al. introduced multiple selection techniques building on the selection using a beam which is aligned to the users hand [65]. All techniques work by calculating the ray from the user's eye-point over an indicated position in space. The *Framing Hands* technique, allows the user to form two corners with the hands and uses the middle of the formed frame as a reference point to draw the selection ray starting from the user's eye point (see figure 2.21). The *Head Crusher*, which is similar to the Framing Hands, indicates the reference point by calculating the trajectory through a point in between the user's forefinger and thumb. The *Sticky Finger* technique on the other hand uses the position of the user's index finger to draw a ray starting from the user's eye point. At last, the *Lifting Palm* technique, uses the position of the outstretched hand and positioning of the palm to cast a ray. The problem using these techniques is the choice which eye's image is used as they display slightly different images. A further problem is the arm fatigue (gorilla-arm-effect), which arises from constantly lifting the hands to the eye-level, though the selection time is reduced in comparison to pointing.

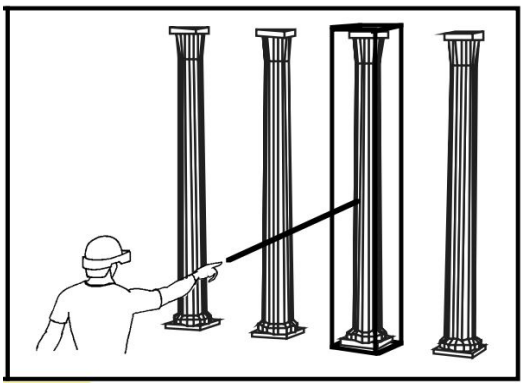


Figure 2.20: Ray casting using a pointing gesture [61].

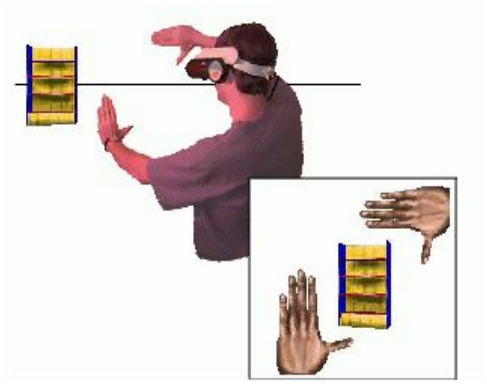


Figure 2.21: The Framing Hands Technique [65].

The *flexible pointer* was introduced by Olwal and Feiner, and extends the existing ray casting selection concepts [63]. Other than previously introduced ray casting techniques, the virtual flexible pointer allows user to select fully or partially occluded objects using a curved arrow as seen in figure 2.22. Olwal et al. noticed that when people point, they usually describe a curved gesture in combination with a movement. Building on this observation, the manipulation of the flexible pointer was based on the hand orientation for the amount of curvature, and the hand position for the length of the selection arrow. This concept expands the selectable objects for the user while making it easier to point on certain objects. The design of the pointer helps to avoid obstruction of view while reducing the need for disambiguation.

The *iSith metaphor* was introduced by Wyss et al. and is also based on a ray cast selection [88]. It enables the movement of two ray pointers, each of which is associated with the movement of a hand. This bi-manual concept allows direct object selection and manipulation while reducing complexity and still granting 6-DOF. The intersection point of the two rays is used to select the virtual objects and provides an easy-to-learn interaction. Such as the previously introduced flexible pointer, it allows the user to grab

partially or totally occluded virtual objects as the intersection of the two rays is essential for the selection and not the first collision with an object (see figure 2.23). The bi-manual concept allows symmetrical and asymmetrical movement while still being limited in the precision of the movement.

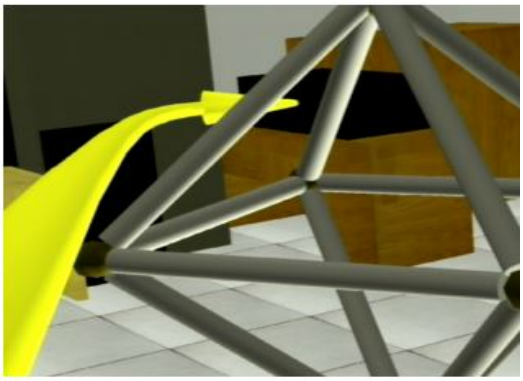


Figure 2.22: Flexible Pointer [63].

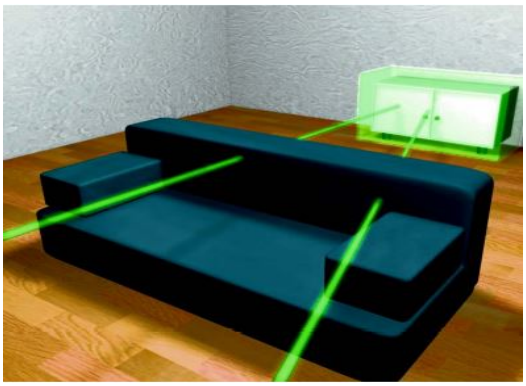


Figure 2.23: iSith ray cast selection [88].

In 2011 Kopper et al. introduced a sphere-casting refinement technique (*SQUAD*) using a four-item menu (*QUAD-menu*) [44]. This technique gradually reduces the choice of selectable objects until the intended object can be selected without the user having to specify. While ray casting amplifies small hand movements which lead to less precise selections due to natural hand tremor and tracker jitter, the progressive refinement suggested is more accurate. Although these techniques require only one precise action, SQUAD has proven to be very accurate and allows for faster selection of small objects.

2.5.2 Manipulation

When examining or exploring an object, the user's goal is to take a closer look as details which may even be hidden. This can be achieved by removing parts of the surface or by using different filters to make these details visible. This subsection introduces several of these techniques.

One of the first publications to investigate the slicing of digital objects, was Osborn and Agogino in 1992 [64]. The *pool of water* metaphor was used to create an interactive and intuitive interface which allowed the direct manipulation of three-dimensional objects. The visualisation was placed within a pool and sliced by the surface of the water within the pool, which represented a cutting plane. The user could change the position of the plate, with the depth of the water pool corresponding to the depth of the cutting plane.

Bier et al. introduced a see-through interface with the concept of tools, which can be made visible on demand (the so called ToolGlass™ widgets) [8]. These contain the visual Magic Lens™ filters, which can be used to reveal hidden information. This filter allows a new style of interaction by providing context-dependent feedback and the ability to view details alongside context. The lens modifies the image in the viewing region and can cull away objects in the local region, while leaving the rest of the view intact. A

further lens is the magnifier which allows the user to rescale the model. A combination of magnifier lens and wireframe lens is displayed in figure 2.24. This see-through interface offers a new design space allowing good graphical feedback while reducing task steps.

Building on the previously introduced Magic Lens™ concept, Viega et al. extended the filter into three dimensions by introducing the visualisation techniques *flat lenses in 3D* and *volumetric lenses* [84]. The flat lenses in 3D allow to preview changes in a limited region, with the visible area being the lens frustum which removes the intersecting surfaces. Flat lenses can also be composed by overlapping them.

Volumetric lenses make it possible to restrict the effect of a lens to a limited volume. This allows to cull away large parts of 3D data sets while still showing the context. An example for the usage of the volumetric lens is *X Ray Vision*, which allows to remove parts of the object within the defined frustum to reveal what's inside as seen in figure 2.25.

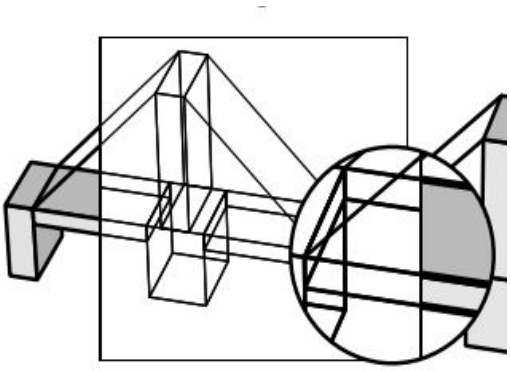


Figure 2.24: 3D wireframe lens and magnifier [8].

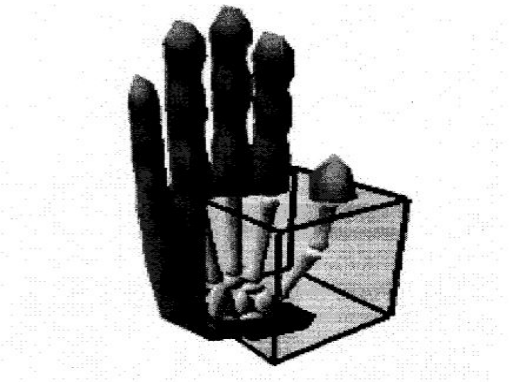


Figure 2.25: X Ray volumetric lens using a defined frustum [84].

Chapter 3

Concept and Design

This chapter deals with the conceptual design of the prototype and serves as a basis for its implementation. First requirements are introduced, which form the basis of the design (see section 3.1), before the overall architecture (see section 3.2), and use cases (see section 3.3) are covered. These are followed by the topic of volumetric data (see section 3.4), the explanation of different interaction techniques (see section 3.5), and how these can be realised in a User Interface (UI) design (see section 3.6).

3.1 Requirements

The prototype is designed to visualise volumetric data in three-dimensional space with an emphasis on the manipulation of the presented data using a touch-sensitive handheld tablet. These interactions should support the exploration and intuitive handling of the data set. Following functions need to be fulfilled by the application:

- Loading both volumetric and iso-surface models must be possible. The latter can hereby be displayed in a three-dimensional space.
- The prototype must offer a selection method to choose a data set visible in a three-dimensional space.
- The prototype must offer multiple methods of exploration. These allow the user to clip away parts of the presented data set and take snapshots.
- The prototype must support the mapping of the displayed three-dimensional model to the input device, which allows the user to move and rotate the data set using a *handheld device* (HHD).
- The system must be designed for minimal direct viewing of the tablet screen, maximising *eyes-free interaction*.
- The prototype must enable the computation of the intersection from the given volumetric data set and the position of the cutting slice within the associated 3D model.

A volumetric data set and a surface data set need to be provided as input for the prototype. The later is rendered in 3D, and since it has no internal structures that can be exposed, the volumetric data set is used to calculate the intersection between the section plane and the model. Further input can be generated through the user with

touch and spatial gestures. The output should be visible through changes in the user's environment and on the model.

3.1.1 Non-Goals

Non-goals were defined in order to specify the scope of the concept and consequently the to be developed prototype more precisely.

- *Remote Connection*
  The focus of this prototype does not lie on the creation of a complex network. Its sole purpose is co-located operation only.
- *Connection to external display*
  Within the scope of the X-PRO project, an external display is often involved in the development and usage of prototypes. In this specific project, this will not be the case at this stage. For example, snapshots will not be viewed on an external display.
- *Multiple Models*
  The prototype is designed for the use of one model at a time, and not for the simultaneous use of several models.

3.2 Architecture

Figure 3.1: Setup as depicted by Mayer et al. [59].

As has become evident in the previous chapter 2, one reasonable choice for this prototype setup is the usage of a *Head Mounted Display* (HMD) in combination with a touch-sensitive handheld tablet as illustrated in figure 3.1. Here a dataset (depicted as a blue skull) is displayed using an HMD and the HHD is used to interact with this dataset. Similar compositions have been implemented by [21, 38, 40, 52, 53, 73, 78].

These setup requirements have been defined based on multiple design insights proposed by Lopez et al. concerning mobile touch systems [53]. A tablet-sized mobile device is favoured as it enables the user to move around the space while supporting 7-degrees of freedom (DOF) navigation (3-DOF translation, 3-DOF rotation, 1-DOF uniform scaling). The mobile display can be used to actively display content, but it can also be used without direct eye contact to the touch interface (eyes-free), to avoid the gorilla-arm-effect [13, 36, 57]. The HHD allows the system to exploit two-handed asymmetric interaction as Guiard has proposed [35]. These favoured asymmetric gestures of the user are executed when one hand is used to hold the device while the other executed touch gestures. The designed gestures are explained in section 3.5 in detail.

The software system components are structured as follows:

User Interface The UI is the visual component and shows the display interaction menu, intersection plane and taken snapshots.

Interaction Logic The interaction logic processes the user input through spatial or touch input, calculates the intersection plane in the model data and enables using the HMD in the virtual space.

Data Input The data input is read from the volumetric data set and used to calculate the section plane. Other inputs are the user's touch and spatial inputs.

Network Manager The network manager is used for the connection and synchronisation between the HMD view and the tablet display, and handles mode and input changes.

The interconnection between these system components can be depicted as seen in the following figure 3.2.

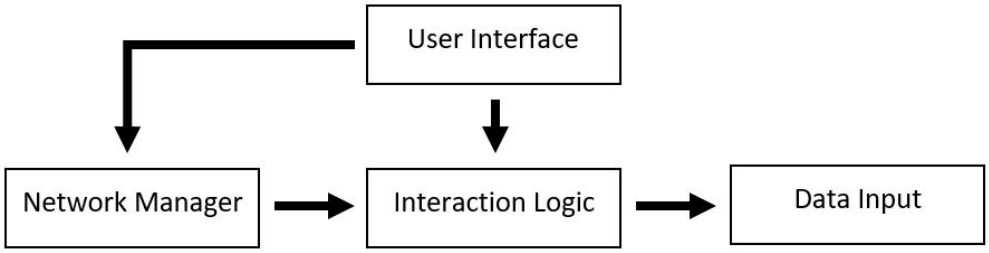


Figure 3.2: Interaction of the components, with arrows indicating dependency.

3.3 Use Cases

The system is expected to allow the user to manipulate and explore a virtual data set. Hereby, the UI is used to show information and enable the interaction with the data set. Following use cases (UC) are to be covered in the prototype using the HHD's touch interface and its spatial capabilities:

UC1 The user moves the rendered virtual model to a suitable position on the room and adjusts the rotation accordingly.

UC2 The user resizes the model to allow an optimal level of exploration.

UC3 The user uses the cutting function to clip away parts of the model and investigates the internal structure.

UC4 The user freezes a cutting plane and can place another one to have a look from multiple perspectives at the model.

UC5 The user saves interesting views by creating snapshots and places them in the environment.

UC6 The user selects a captured slice, inspects it and its neighbours.

UC7 The user removes all snapshots from the virtual environment.

UC8 The user resets the model.

3.4 Volumetric Data Sets

Volumetric data is defined as a set of samples which represent a value at a three-dimensional location [43]. A sample can be defined as (x, y, z, v), whereas x, y, and z indicate the cartesian position of the value v. The given value v can be binary or multi-valued, meaning it describes certain properties of the given data such as its colour or density.

3.4.1 Computer Tomography

Such three-dimensional data can be obtained as a sequence of 2D images using *Magnetic Resonance Imaging* (MRI) or *Computer Tomography* (CT). While CT is one of the most accurate methods of image generation because thin image slices increase the accuracy of the imaging, MRI uses thicker layers and therefore cannot achieve such high accuracy [82]. MRI uses non-ionising radio frequency radiation and measures the magnetic resonance of hydrogen molecules, which makes it more useful for recording soft biological tissue [28]. CT, on the other hand, emits an X-ray beam and measures its attenuation from multiple directions. It can scan any surface or material up to a certain thickness and allows for a high density of information. The absorption of the X-ray shows the density of the material which is scanned, though the X-ray photon energy must be adjusted to penetrate thick objects or very dense material. An example for a slice of a CT scan is shown in the lower part of figure 3.3 [1]. It shows how denser material, such as the neck and head of the guitar, absorbs more rays and is therefore shown in a lighter colour than thinner parts as seen on the wooden guitar body.

Figure 3.3: CT Scan of a concert guitar done by *Research Group Computed Tomography*.

When the scanned object consists of multiple materials with different density, the scan might be difficult to execute as weaker materials cannot properly be recognised [28]. To allow the correct scan of such objects appropriate thresholds need to be identified

[1] https://www.3dct.at/cms2/index.php/en/ct/fields-of-application/140-special-applications

and applied. As CT scans are a non-destructive method, they are used in industry to inspect external and internal geometry of complex objects to detect flaws.

3.4.2 Rendering Techniques

Volumetric data is difficult to process, as the data sets are extensive and thus require large memory capacities, and computing power. Therefore, they are often converted into a simpler form to allow easier processing and display, faster calculations, and lower memory requirements. [51, 75]. One way to reduce the size of a volumetric dataset is to use a surface rendering technique [86]. Such a technique is used to approximate the surface of the data set using geometric primitives, which can then be rendered.

A prominent method for extracting and rendering such surfaces is the *Marching Cubes Algorithm* [54]. This algorithm uses an interpolation function to intersect edges of the data in order to create triangle models. The defined surface then consists of multiple triangles and is called iso-surface. The Marching Cubes Algorithm uses 14 topologies to differentiate between different parts of the layer. Each of these 14 cases represents a generic set of triangles and thus allow to approximate any part of a surface.

The *Marching Tetrahedra Algorithm* is similar to the per cell approach of the Marching Cubes Algorithm, which marches a cube through the data set and creates faces by interpolating between vertices [25]. In contrast to the Marching Cubes, Marching Tetrahedra splits this marching cube into six irregular tetrahedra which allows to calculate intersections with nineteen instead of the twelve edges.

A mayor drawback of the surface technique, is the fact that the surface of the data set can only be approximated if geometric primitives are used, while the data below the surface is lost. Alternatively, volume rendering techniques are used. These create 2D images directly from 3D volumetric data without the intermediate step of calculating geometric primitives [86]. The inner structure of the data is not lost, though the size can not be reduced as much as with the former method.

3.4.3 Prototype Data Sets

The prototype must be able to represent the surface of the three-dimensional object, but also have the ability to retrieve parts of the object's internal structure. It therefore requires two separate data sets, one reflecting the volumetric data of the original dataset, and a surface dataset representing a simplified surface:

Volumetric data set It contains all data acquired from the real world object and is only needed when referring to the internal structure of the model. It is used, for example, to calculate cutting surfaces, which can then be superimposed when clipping the simplified model to visualise the complex internal structure.

Surface data set It is a simplified version of the original data set. A surface rendering technique is optimal in this case as it enables the extraction of an approximated surface model and leaves out the internal structure, which allows for a sizeable reduction in calculation time when rendering the data in a three-dimensional space.

3.5 Interaction

As already covered by Mayer et al. [59], there are various intuitive ways of utilising a touch-sensitive mobile device to investigate three-dimensional data in a *Mixed Reality Environment* (MRE). Figure 3.4 shows how such inputs could be connected and put to use. The starting point of the interaction concept is the main menu. From there, either explorations or selection functions can be started, or changes to the model and the user's environment can be undone. In exploration mode, the user can perform inspection methods with tap, swipe, and grab gestures. Selection mode, on the other hand, allows the user to select an object, switching to selection mode, and adjust and reposition it by shaking, tilting, touching, rotating, and pinching the HHD. The different input possibilities and their structure are explained in detail in subsection 3.6.

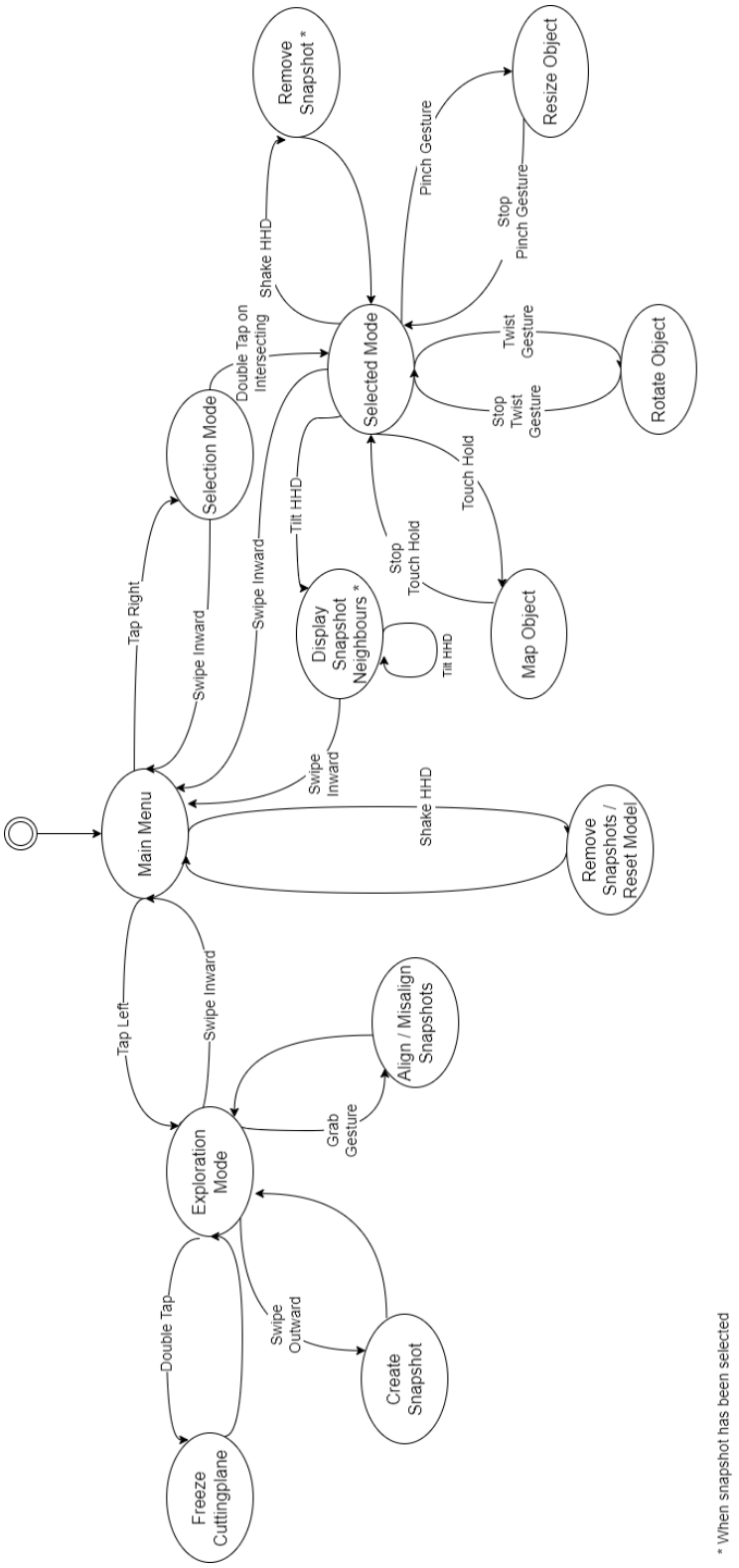


Figure 3.4: Diagram describing the different interaction states.

3.5.1 Navigation

Lopez et al. state in their guidelines that users should have adequate means of navigation and be able to move around the room freely [53]. Since the model is placed in the user's real environment, the user can freely move around the virtually presented object and view it from different perspectives.

3.5.2 Spatial Transformation

According to the mentioned design guidelines [53], the prototype should support 7-DOF (3-DOF translation, 3-DOF rotation, 1-DOF uniform scaling). These manipulations are covered by the following methods for mapping, rotating and resizing the presented data.

Mapping

6-DOF interaction is allowed by mapping the position and rotation of the mobile device to the virtual model. This way the object always has the same distance to the HHD and follows the user to any given tracked space. In case the user wants to change the distance to the object when mapping translation and rotation, the mapping mode has to be restarted.

Rotation

In addition to the mapping option, the user has the possibility to explicitly rotate a selected object by creating a rotational gesture on the tablet. This touch input enables better performance in making large orientation changes than pure spatial input (such as mapping), which is much more difficult to execute holding a tablet [58]. The touch gesture requires at least two fingers to move on the touchscreen as seen in figure 3.5. As Cohé et al. noted, users may use more fingers than necessary to perform a touch gesture, so users should also be enabled to do so for this input [26]. The rotational input allows the object to be yawed (rotated around the z-axis) when the touch-surface is held horizontally, and rolled or pitched (rotated around the x- or y-axis) when the device is held vertically. If the rotation is executed in a roll or pitch depends on the spatial relation of the tablet to the respective object.

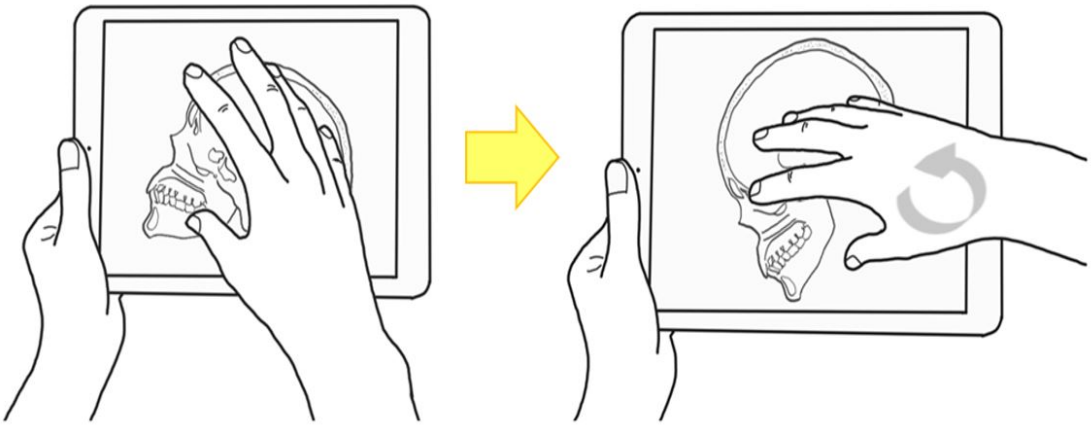


Figure 3.5: Rotations can be executed using a twist gesture as depicted by Mayer et al. [59].

Resizing

A virtual object or any chosen view can be resized by performing the common pinch gesture on the touch interface of the HHD as shown in figure 3.6. This has also been found to be the most expected gesture for this interaction in a formative study conducted by Surale et al. [78].

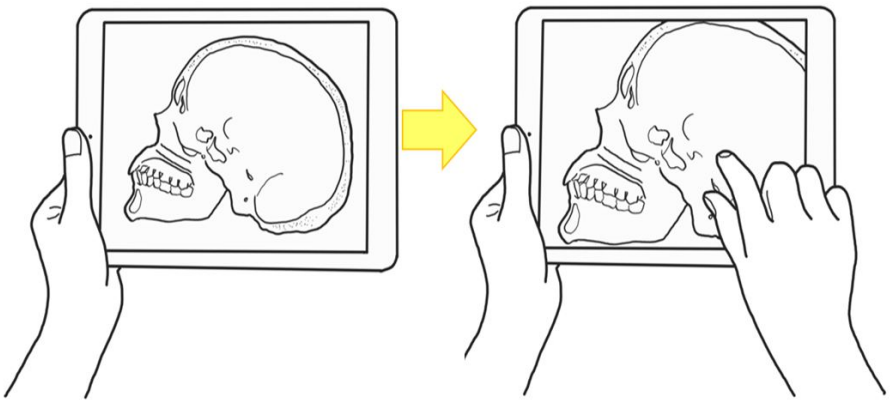


Figure 3.6: The resizing of an object can be executed using a pinch gesture as depicted by Mayer et al. [59].

3.5.3 Selection

There are multiple selection techniques which can be differentiated using several characteristics. For fulfilling the requirements of this prototype, the taxonomy created by Besançon et al. [5] (introduced in chapter 2) was used to choose a suitable selection technique. Taking into account that the user of this prototype has to execute basic single-

object selection tasks without shape creation and adjustment, a simple ray metaphor suffices.

Ray metaphors have the disadvantage of requiring the user to have a clear view of the target. When following the taxonomy of Besançon et al., these characteristic decisions leave the choice between Ray-Casting [61], Framing Hands [65], iSith [88], Flexible Pointer [63] and SQUAD [44]. All of these techniques have at least 5-DOF for the selection.

As the selection technique would best be executed using a tablet and avoiding additional hardware (apart from an HMD and an HHD), bi-manual selections such as Framing Hands and iSith are not suitable since the tracking of the hands is necessary to execute these selection techniques even though Framing Hands allows a faster selection and iSith even allows the selection of partially or totally occluded objects. The flexible pointer must also track the user's hand and arm movements and is therefore also not suitable for the prototype. Although SQUAD selection is superior to conventional ray cast selection in terms of accuracy, this is not necessary due to the intended scope of the user's virtual environment, which is to cover only the model and placed snapshots in the user's immediate vicinity. Further, SQUAD would complicate the selection of placed snapshots compared to the simple ray cast selection. Bowman advises to rely on a ray casting technique if speed is a requirement making ray cast selection the appropriate technique for this prototype [17]. This technique works best with a posture in which the arms are slightly bent. According to Hincapie et al. this position is the least strenuous for the user's shoulders and arms in terms of the gorilla-arm-effect [36].

To provide the user with an alternative selection option in case of occlusions, the list selection proposed by Mine shall also be included as an indirect selection technique in the prototype [61], similar to the slice gallery as implemented by Luo et al. [55].

3.5.4 Clipping

The removal or clipping of parts of an virtual object gives the user a new perspective and provides a more thorough understanding of its internal structures. As previously mentioned in chapter 2, the exploration of data sets with the help of cutting planes has been implemented in multiple publications [15, 22, 33, 40, 67, 71, 76, 78, 80, 86]. While most of those prototypes utilised a handheld device to interact with the presented data set, only a fraction made use of a mobile tablet with touch-capabilities [40, 78].

Malizia et al. talk about the artificial naturality, which is currently conventional in the design of natural interfaces [56]. Hereby they refer to the fact that specific gestures must be learned rather than relying on familiar gestures used when interacting with real world objects. When using a handheld tablet as a direct mean of cutting and displaying the internal structure as the given point, we mean to break this artificial naturality and cross over to intuitive, real naturality by allowing the user to interact with the object as naturally as possible.

The HMD visualises the surface model in an virtual environment while the HHD can be used to cut away parts of the data set. A clipping plane is positioned slightly above the display and any part through which the HHD passes through is removed. Simultaneously, the display is used to show the internal structure of the original data at the exact position of the device within the model. Hereby the described naturality

can be achieved as this concept allows the user to use the tablet as a type of window to physically and virtually look into the displayed model.

As the clipping plane is positioned slightly above the display, the boundary of the surface model will still be visible between the device and the virtual cutting plane (see figure 3.7). Such a logic has already been implemented in a prototype of Bornik et al. [15] which serves as an inspiration. This allows a more accurate evaluation using object boundaries while the display shows the original volumetric data at the position of the HHD.

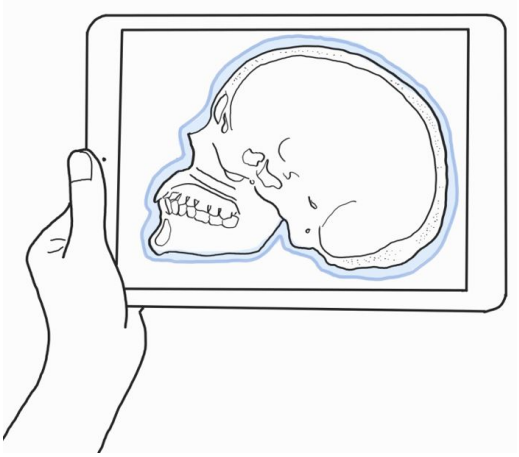


Figure 3.7: The calculated volumetric cut is displayed on the tablet while the virtual boundary of the virtual object (blue, visualised using the HMD) is still visible.

The position of the clipping plane can be frozen within the model by double tapping the touch interface. This enables multi-planar slicing as several planes can be set within the model, similar to Fröhlich and Plate, who implemented three cutting planes which could be manipulated using the cubic mouse [33]. In place of the frozen cutting plane the previously calculated slice of volumetric data is rendered to allow an inspection without the creation of a snapshot. This can be seen in the following figure 3.8, where the calculated internal structure is superimposed on the clipped part of the surface model. As the HMD might not allow a satisfactory view on the tablet's surface, the intersection plane will be virtually superimposed in form of a monoscopic view over the HHD.

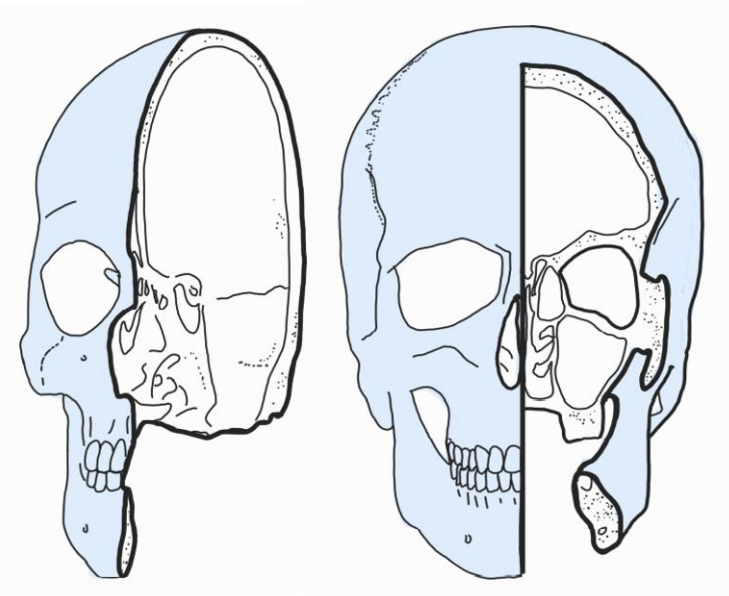


Figure 3.8: Clipped virtual surface model (blue) with frozen cutting planes displaying the volumetric data superimposed on the clipped part.

3.5.5 Snapshots

When the mobile tablet is positioned within the surface model, the internal structure can be calculated at this position from the original volumetric data set and displayed on the device's touch interface. This calculated slice can be saved by taking a snapshot. The computation could potentially be intensive, so audio and visual cues are used to inform the user that the calculation of the captured slice is in progress, thus confirming the action [24, 37]. Such a snapshot can be created by placing it within the user's environment using a swiping gesture. This gesture can be designed naturally and compared to tipping a ball or small object into a new place. The swipe direction, such as a flicking direction of a ball, is used to calculate where the snapshot should be placed in the user's environment (see figure 3.9). The snapshot is rendered as a visualisation in AR, similar to the virtual displays introduced in subsection 2.3 [62, 66].

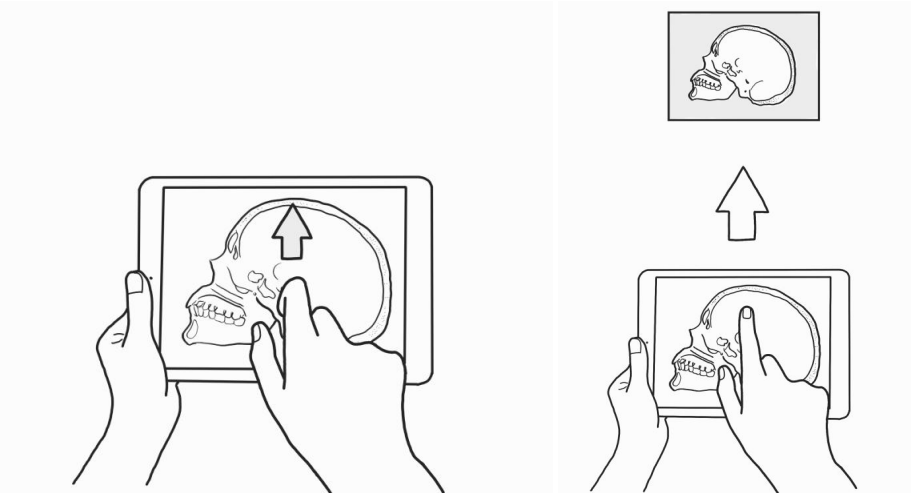


Figure 3.9: Snapshots are a momentary recording of a cutting slice and can be placed in the user's environment. The swipe direction tells the system in which direction the snapshot should be placed.

A captured slice can be moved around the user's environment, rotated, resized, or removed. The position of the cutting plane where the snapshot was taken will be saved, so the connection can be queried. To allow the user a view of the snapshots in the virtual environment, the position of the tablet is used to align the orientation of the snapshot views with the position of the user (such as Hubenschmid et al. implemented in [38]). This proxemic method allows the user to move around the environment while simultaneously always being able to view the placed images.

Alignment

As shown by Normand et al., the extension of a physical display by means of virtual views is well accepted by users and can lead to an increased performance [62]. With this in mind, the prototype should allow the user to align all taken snapshots around the device by performing a grab gesture on the touchscreen (see figure 3.10). This allows for a quick overview without requiring the user to look around the entire environment where the snapshots would otherwise be placed.

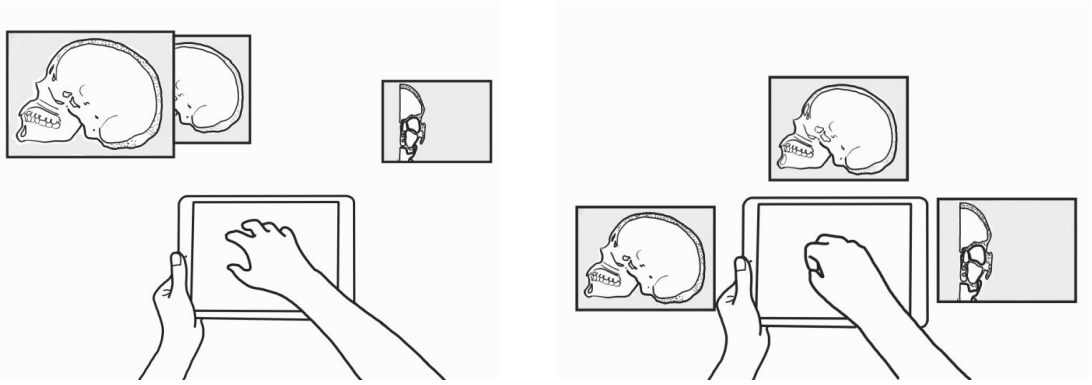


Figure 3.10: All snapshots are placed at specified locations in the user's environment. Occlusion is possible. The grab gesture triggers the alignment of existing snapshots around the HHD.

Neighbour Views

Yang et al. presented the Tilt Map, a novel interaction technique which enables switching between 2D and 3D visualisations by combining an HMD with a handheld controller [89]. Hereby the tilt angle of the controller was used to switch between different views. A conducted study showed that the tilt control was statistically more accurate without a significant cost in time when compared to traditional techniques such as a side-by-side comparison.

Following this publication and the recommendation to incorporate spatial triggers by Hubenschmid et al. [38], we incorporate an intuitive tilting movement of the mobile device to switch between a taken snapshot and its neighbouring slices. This interaction possibility allows to inspect the data set along the z-axis next to the selected snapshot. If tilted to the left, the clipping plane is moved forward on the z-axis, displaying the internal structure of the model on this side of the snapshot. When the device is tipped in the opposite direction, the calculation continues backwards on the z-axis. Boring et al. propose to use a snapping behaviour when implementing a tilt movement [13]. Therefore the user will be able to snap from one cutting plane to the next position instead of a fluent transition which would be computationally more expensive.

3.6 User Interface

A publication describing a similar design and concept has been published by Surale et al. in 2019 [78]. Other than this prototype, which was developed for the inspection of volumetric data, they focus on 3D solid modelling. They found that holding the tablet in the non-dominant hand can be tedious for the user, which is in line with the guideline of Lopez et al. stating that holding the HHD up should be avoided as often as possible, otherwise fatigue problems arise. Therefore, the UI of this prototype will be designed to prioritise eyes-free interaction, which has also been proposed by Hubenschmid et al. to guarantee a practical interaction [38]. This eyes-free menu is implemented in such a way that touch commands are as simple and intuitive as possible, and the current mode

of the application is clearly indicated by the objects or by an additional *head up display* (HUD). The HHD should therefore only need to be raised when it is used to intersect the model, to select or rotate an object, or to examine a snapshot and its neighbours. In addition to visual feedback given over the HHD's display or a superimposed virtual view, the system can also give the user haptic and auditory feedback [24, 37]. The user can directly interact with the system using the touch-display and entering gestures directly on its surface or through spatial gestures by moving the device.

3.6.1 Main Menu

The interaction methods mentioned in section 3.5 can be summarised in the categories selection and exploration, which allows the main menu to be designed bi-sectional, offering one of these two as an option (see figure 3.11). The UI is divided into two equally sized parts, one to enter the selection mode and one to enter the exploration mode. As these buttons cover the entire surface, the user only needs to know which option is on which half of the screen in order to make a selection. This allows the user to make an eyes-free mode selection.

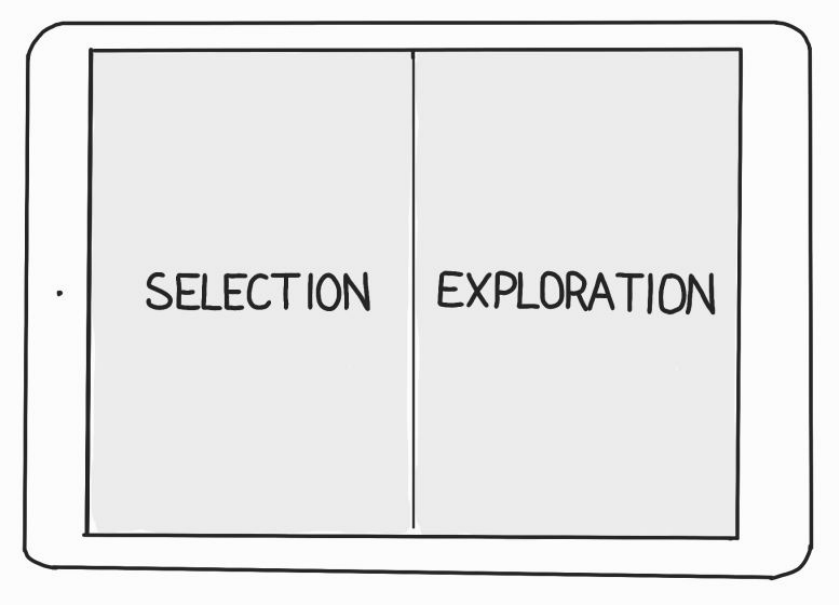


Figure 3.11: Bi-sectional menu parted into selection and exploration mode.

To reset all changes made, the HHD can be shaken multiple times in the main menu. Shaking first removes all displayed snapshots in the user's vicinity. If the user then shakes again, the model is reset and returns to its original state, removing all frozen cutting planes. All actions possible in the main menu are listed below in table 3.1

Action	Beforehand	Description
Start Selection Mode	-	Tap the left side of the screen (selection part).
Start Exploration Mode	-	Tap the right side of the screen (exploration part).
Remove all snapshots	Captured snapshots	All snapshots can be removed by shaking the HHD multiple times without a previous selection.
Reset object	Clipped object, no snapshots	The virtual object can be reset to its original state by shaking the tablet multiple times.

Table 3.1: Available actions in the main menu.

3.6.2 Selection Mode

The user can enter the selection mode when tapping the left part of the main screen (selection part). This way the ray cast selection (described in 3.5.3) is activated. When the ray hits an object, the user can use a double tap on the UI to confirm the selection. If the selection was successful, the application should enter the selected mode. The selected mode allows the user to execute spatial mapping (translation and rotation as described in section 3.5.2) of the selected object while at least one finger is continuously touching the screen. As soon as the touch stops, the mapping also stops. While an object is selected, the touch interface can be used to perform a pinching gesture to resize the selected object. The mode and selection can be terminated using an inward swipe as is conventional with modern smartphones. As an additional spatial interaction, the user has the choice to inspect neighbouring slices by tilting the tablet. An additional menu can be opened by using three-fingers to swipe down the tablet's surface. This menu contains all taken snapshots and allows their individual selection. These interaction options are listed in the following table 3.2.

Action	Beforehand	Description
Select object with ray	-	Point ray at object and double tap to confirm selection.
Select object with list	-	A grab gesture opens the object selection list. An object can be selected by taping on the object in the list.
Map object	Selected object	As long as a finger continuously touches the screen, the selected object should mimic the motion (translation and rotation) of the HHD.
Rotate object	Selected object	Additonal to mapping, a rotation can also be executed by performing a twist gesture on the touch-sensitive surface using at least two fingers.
Resize object	Selected object	The size of an object can be adjusted by performing a pinching gesture on the screen.
Investigate neighbour slices	Selected snapshot	Neighbour slices of a snapshot can be investigated by tilting the tablet to the left or right to see the slices further along the z-axis.
Remove snapshot	Selected snapshot	A selected snapshot can be removed by shaking the HHD once.
End mode	-	Performing a swipe-in gesture from the side of the handheld tablet closes any mode and deselects all objects.

Table 3.2: Available actions in selection mode.

3.6.3 Exploration Mode

The exploration mode is entered by tapping the right part of the main screen (exploration part). In this case a clipping plane is automatically aligned with the HHD. The user can start inspecting the internal structure of the model displayed using the HMD by physically moving the HHD, and thus also the clipping plane, through the presented data.

The clipping of the data happens as soon as the model is intersected with the cutting plane, though the internal structure is only calculated after holding the HHD in place for a short period of time. A double tap on the interface allows the user to freeze the cutting plane. In this case, a new clipping plane is automatically created so the user has the option to set multiple planes (see subsection 3.5.4). When inspecting the

internal content of the data, a swipe on the interface allows the creation and placement of a snapshot (as described in subsection 3.5.5). A line connects the snapshot with the cutting plane placed in the exact position where it was taken. The user can exit the exploration mode using a simple swipe-in gesture on the touch-sensitive surface of the HHD. The following table 3.3 gives an overview over the introduced interaction options which are usable in the exploration mode.

Action	**Beforehand**	**Description**
Clip object	-	Moving the clipping plane mapped to the HHD through the model to remove the parts that have been passed through.
Freeze clipping plane	Clipped object	The cutting plane can be placed within the model by double tapping the screen when it is positioned within the model.
Capture snapshot	Clipped object	An outward swipe into any direction gives the direction in which a snapshot should be placed. A snapshot is created when it is placed in the users environment.
Align snapshots	Captured snapshot	All snapshot can be aligned around the HHD by performing a grab gesture on the touchscreen.
Remove all snapshots	Captured snapshots	All snapshots can be removed by shaking the HHD once without a previous selection.
Reset object	Clipped object	The virtual object can be reset to its original state by shaking the tablet multiple times.
End mode	-	Performing a swipe in gesture from the side of the handheld tablet multiple times.

Table 3.3: Available actions in exploration mode.

Chapter 4

Implementation

This chapter gives an insight into the implementation of the prototypical application created as part of this master thesis. First, section 4.1 describes the technologies used, before section 4.2 gives an overview of the different types of data required. Section 4.3 describes the details of the implementation before the chapter concludes with the limitations faced during the development (see section 4.4).

4.1 Technology

This section introduces the necessary hardware (subsection 4.1.1) and software (subsection 4.1.2) for the created prototype. The described hardware was used during most of the implementation process, as well as for the preliminary study described in chapter 5.

4.1.1 Hardware

The prototype is composed of several different computational devices. A PC and a connected *Head Mounted Display* (HMD) handle the *Augmented Reality* (AR) application together with the main computing tasks. The user input for the system is provided via a tablet and a tracking marker. The relationship between these devices and the user is illustrated in figure 4.1.

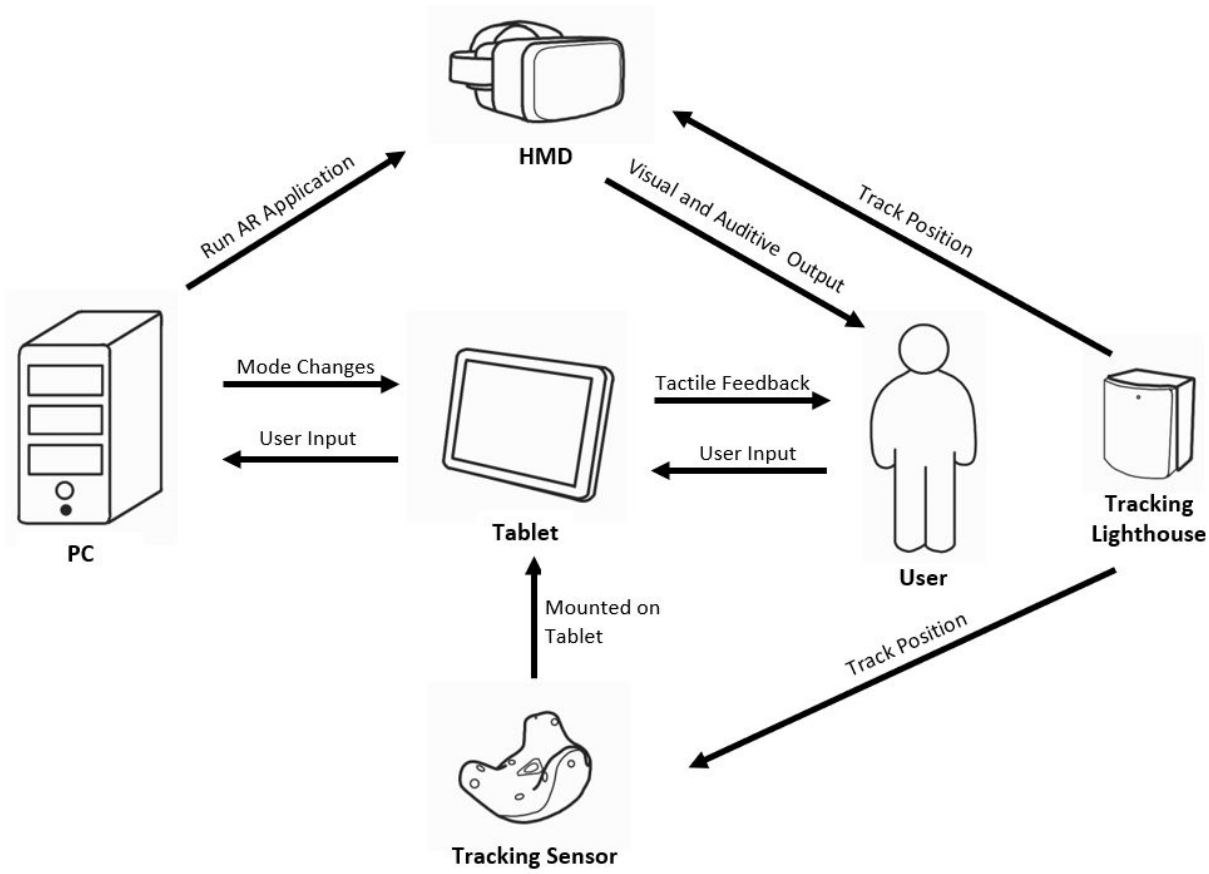


Figure 4.1: Interconnection between hardware devices and user.

HMD

This application uses an HTC Vive Pro Eye[1] HMD with a Dual OLED 3.5 inch diagonal display which provides 1440 x 1600 pixels per eye. It has a refresh rate of 90 Hz and a *field of view* (FOV) of up to 110 degrees. It is equipped with tracking sensors, earphones, a microphone, and front facing cameras which enable the usage of AR.

The HMD is paired with a SteamVR Base Station Tracking 2.0[2], which has a 120 degree FOV. These devices are used to pick up the tracking markers in a defined area and keep track of changes to their position and rotation. Apart from an HMD, VIVE trackers can be used to mark objects and places and keep track of their position in the augmented or virtual environment.

[1] https://www.vive.com/us/product/vive-pro-eye/overview/
[2] https://www.vive.com/us/accessory/base-station2/

PC

The PC is equipped with 16.0 GB RAM, an Intel®[3] Core(TM) i7-10750H (2.6 GHz) CPU, a NVIDIA[4] GeForce RTX 2070 with Max-Q Design GPU and used the Intel® Wi-Fi 6 AX201 (160 MHz) Wi-Fi adapter.

Tablet

As *handheld device* (HHD) a 10.36 inch tablet running Android 11 is used. It weighs 499g and is equipped with a Samsung Celeron 3865U 1GHz processor and 4 GB RAM. The device contains a Wi-Fi adapter, an accelerator, a linear accelerator, a gravity sensor, as well as a gyro sensor.

Tracking Marker

A HTC tracking marker is fitted to the tablet to keep track of its position within the user's environment. The tracking sensor is mounted on a camera quick shoe, while the tablet is framed by a tablet clip holder. The arm of the clip holder can be rotated and adjusted to see fit for usage. The holder and the quick shoe are connected using a specifically 3D printed connection part which can be seen in figure 4.2 and figure 4.3. It is designed in a way that allows for the quick removal of the tracking device to charge it.

Figure 4.2: The clip holder is mounted on the back of the tablet, only touching the top and bottom.

Figure 4.3: The connection part (white) connects tracking marker and tablet clip holder.

Tracking Space

The prototypical application allows the user to operate in AR. Apart from the above listed hardware no more interaction objects are needed. Therefore, an empty environment which allows the user free, unconstrained movement is sufficient.

[3] https://www.intel.co.uk/content/www/uk/en/homepage.html
[4] https://www.nvidia.com/en-gb/

4.1.2 Software

The system was implemented using a game development platform together with plugins.

Unity

The game engine *Unity*[5], more specifically *Unity 3D* (version 2019.2.21f1), is used to implement both the AR and the Android application. This real-time development platform provides a physics engine as well as lighting and audio. The system's behaviour can be defined using the provided C# scripting API. The framework operates cross-platform, is widely used, and its asset store offers several plugins. Projects can be enriched by using such plugins to import objects, materials, shaders, and logical components. They are also used to support *Mixed Reality* (MR).

SteamVR

SteamVR [6] (version 1.22.13) is used to power VR and AR, and is needed to connect an HMD to a PC. It allows the user to setup the interaction area and serves to manage the devices and their status. Additionally, a SteamVR unity plugin[7] is integrated to represent the *Player* prefab in the AR scene.

SRWorks

The *SRWorks SDK*[8] (version 0.9.7.1) is provided by HTC Coporation and enables the see-through stereo camera view for HMDs such as the VIVE Pro or VIVE Pro Eye. Access to the front-facing stereo cameras allows the display of depth, spatial mapping, and the placement of virtual objects in the foreground or background. The prototype requires this see-through functionality to enable AR for the HMD. For this to work, The SRWorks plugin must be imported to the Unity application and the *SRwork_FrameWork* prefab needs to be added to the respective AR scene.

4.2 Data

The prototype uses different formats of the same data set to allow its rendering and exploration. Figure 4.4 shows how the data is processed in order to be used for the prototypical implementation. First, the real world object[9] is scanned using a CT scanner. For the volumetric information, the data is then exported in form of two-dimensional image slices, as explained in subsection 4.2.1. For the surface data, the information is first transformed into iso-surfaces, before being exported as a mesh and simplified in order to be rendered in AR, as described in subsection 4.2.2.

[5] https://unity.com/
[6] https://www.steamvr.com/en/
[7] https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647
[8] https://developer.vive.com/resources/vive-sense/srworks-sdk/
[9] https://www.howtoheatpress.com/wp-content/uploads/2020/12/Best-Handheld-Sewing-Machine.jpg.webp

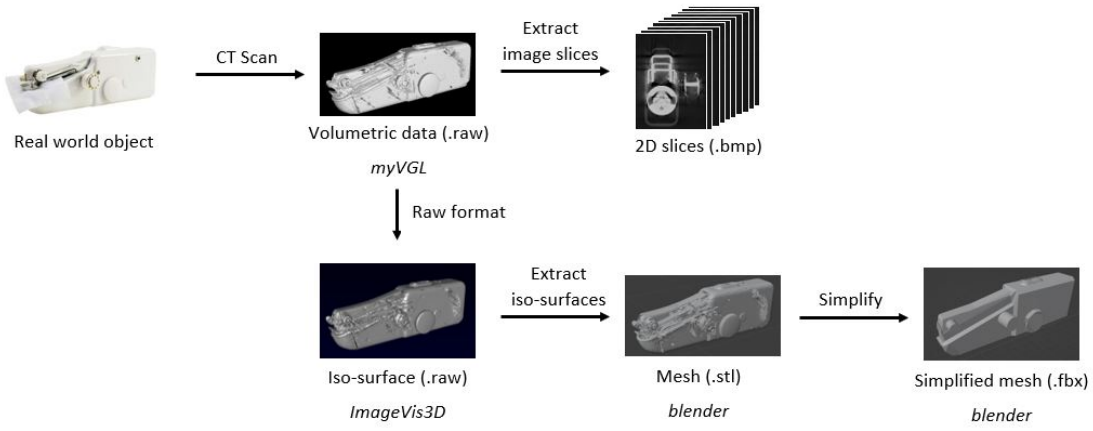


Figure 4.4: Data processing for its use for the prototype.

4.2.1 Volumetric Data

The data used for this prototype was created by the *CT Research Group*[10] at the Wels Campus of the Fachhochschule Oberösterreich. A handheld sewing machine was scanned using the 225kV tube of a RayScan 150E[11] CT system. The collected volumetric data was then exported in raw format with a configured *region of interest* (ROI) in which the size and material type of the sewing machine was tailored.

This volumetric data (see subsection 3.4) was inspected using *myVGL* 3.4[12], which is a viewer application for three-dimensional data. It can be used to inspect the data set and read information such as the resolution listed in table 4.1. First, the data was inspected, then exported along the z-axis using captures every 0.2 mm resulting in 1,101 slices.

	x	y	z
voxel dimension	453	653	1,767
dimension μm	56,407.1	81,310.91	220,025.1
resolution μm	124.5	124.5	124.5

Table 4.1: Resolution of the volumetric dataset.

MyVGL uses a left handed coordinate system; images how the data looks at specific points can be seen in the following figures 4.5, 4.6, and 4.7.

[10] https://www.3dct.at/cms2/index.php/en/
[11] https://www.3dct.at/cms2/index.php/en/equipment/96-rayscan-en
[12] https://www.volumegraphics.com/en/products/myvgl.html

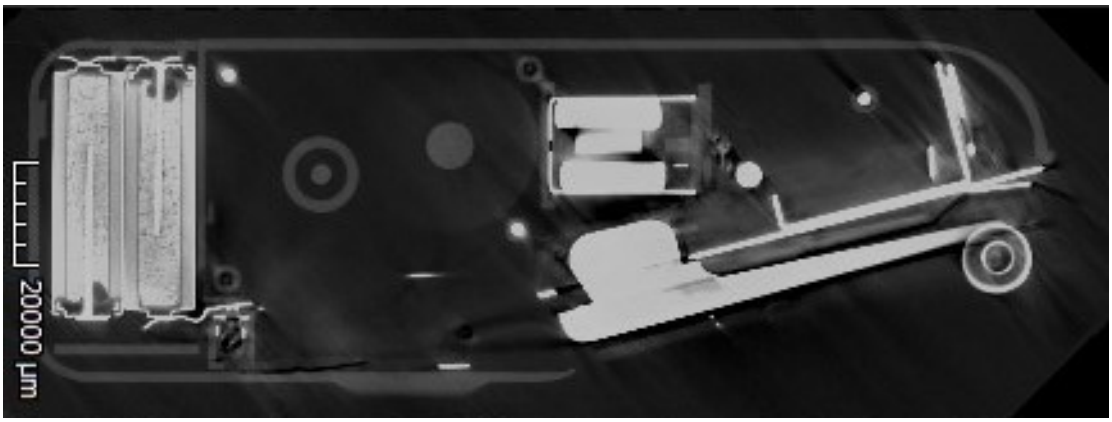


Figure 4.5: Image of a slice along the x-axis of the volumetric data.

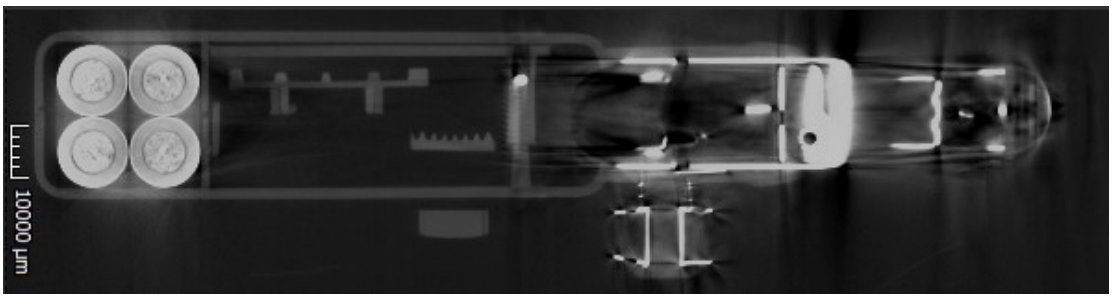


Figure 4.6: Image of a slice along the y-axis of the volumetric data.

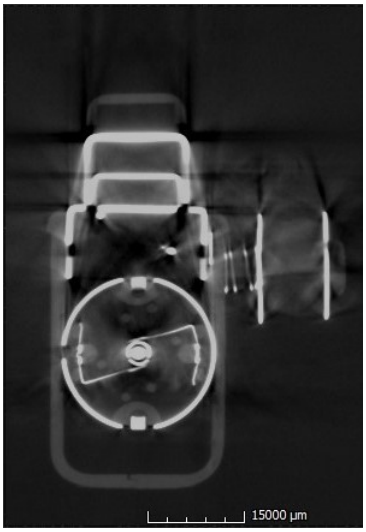


Figure 4.7: Image of a slice along the z-axis of the volumetric data.

The sewing machine consists of different materials which differ greatly in their density. The combination of different materials in an object can cause difficulties when recording a CT scan, as described in the subsection 3.4.1. As can be seen very clearly in the images above, materials such as steel, for instance in area of the batteries on the left side of figure 4.5 and figure 4.6, strongly absorb the irradiated x-rays. While they are recorded in a very bright colour, the surrounding material, which has a lower density, is not captured as clearly as it is when the rays are not disturbed by higher density materials. This difference in material and density causes parts of the scan to be less sharp than others and the contours to blur.

4.2.2 Surface Data

For the three-dimensional model, the same raw file was imported to *ImageVis3D*[13] (version 3.1.0) to create an stl file (Standard Triangle Language), which are commonly used to describe the geometry of 3D surfaces. ImageVis3D is a volume rendering program which offers iso-surface settings to adjust the iso-value for rendering. The iso-value is a scalar value which is used to generate an iso-surface by stating which density values are to be extracted. Since the sewing machine consists of several materials of different densities, it is difficult to set a good boundary (iso-value) which separates the wanted from the unwanted values. In any case, there is always one material that cannot be properly distinguished. Figure 4.8 and figure 4.9 show how the object looks like when the boundary is set too low or too high.

Figure 4.8: Object with the iso-value chosen too low (12,533) showing disturbances on the surface.

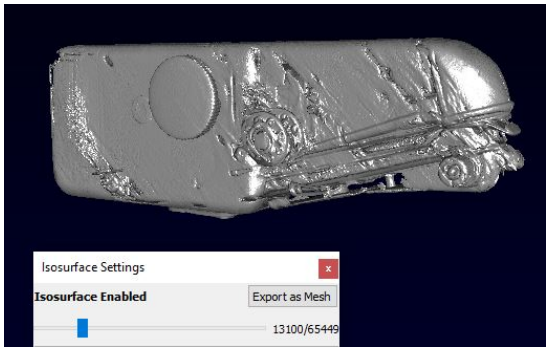


Figure 4.9: Object with the iso-value chosen too high (13,100) showing many holes on the surface.

Due to the interference caused by the different densities of the materials, the model must be manually adjusted to fill holes and smooth surfaces. When the boundary is set too high, the model contains too many holes, while setting the boundary too low results in too many disturbances in the model's surface. The boundary should therefore be chosen in a way to make the iso-surface as similar to the original as possible. The dataset shown in figure 4.10 was rendered with an iso-value of about 12,900, which gave the best result for all captured materials.

[13] https://www.sci.utah.edu/software/imagevis3d.html

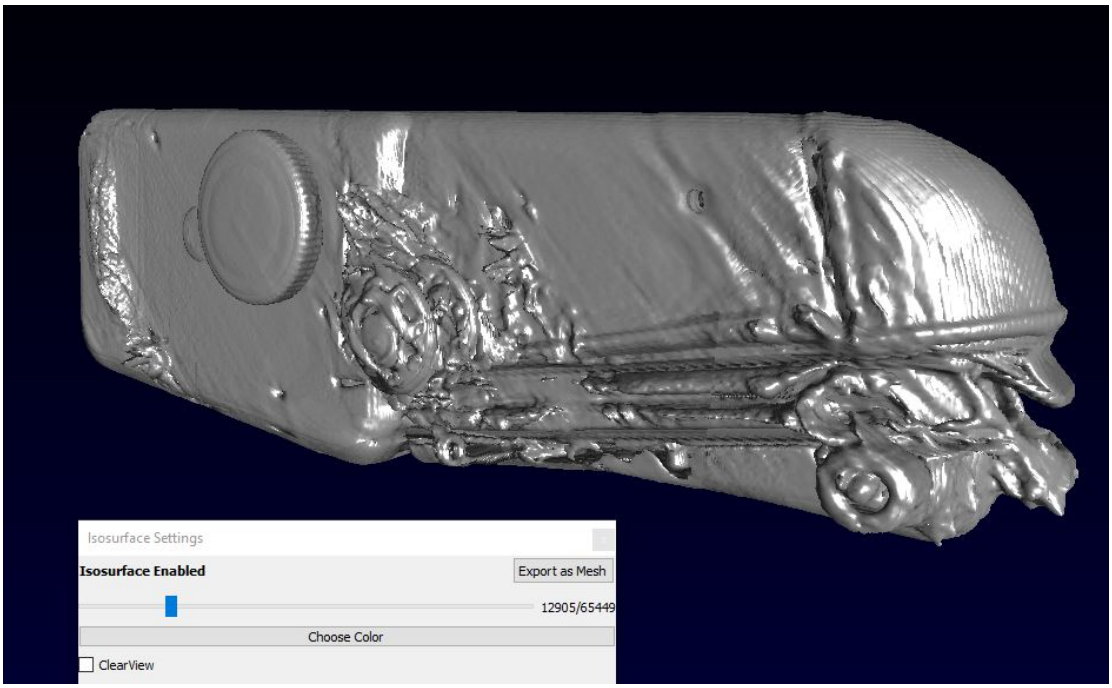


Figure 4.10: Selection of number of iso-surfaces using ImageVis3D.

The figure perfectly visualises how the combination of materials having different densities impacts the result of a CT scan. In the back of the image on the left side, there is a disturbance of the surface (original material plastic, low density) caused by the batteries (original material carbon, lithium, and other metals, high density), located within. The plastic cover area around the stainless steel coil in the middle of the machine is also disturbed, which is most likely caused by both the coil and the motor of the sewing machine.

The mesh was exported from ImageVis3D in form of an stl file. Due to the disruptions, the data could not be used as the final three-dimensional model to be displayed in the prototype. In a next step, the open source 3D content-creation program *blender*[14] (version 3.2.0) was used to simplify the data and apply manual adjustments. It showed that the extracted data contained 6,158,543 faces, which needed to be simplified to make it easier to work with the data set. Blender's decimate feature was used to reduce the number of faces to 307,927, 5% of its original size.

As the surface of the model contained holes and showed great disruptions, it was further edited manually. Faulty surfaces were replaced with planar surfaces, and leftovers of the internal structure were removed completely. Removing the data below the surface was necessary to properly render intersection planes on cut parts of the 3D object in Unity. The simplified surface dataset was then exported as a mesh in form of an fbx file. Fbx allows the usage of 3D models in multiple digital modelling programs, such as Unity. It works with binaries and is therefore able to load data sets fast. The result of the manual adjustments compared to the original with the deformed surfaces can

[14] https://www.blender.org

be seen in figure 4.11. The left side holds the imported mesh with all disruptions and holes. As can be seen in the figures on the right side, almost all surfaces needed manual adjustment.

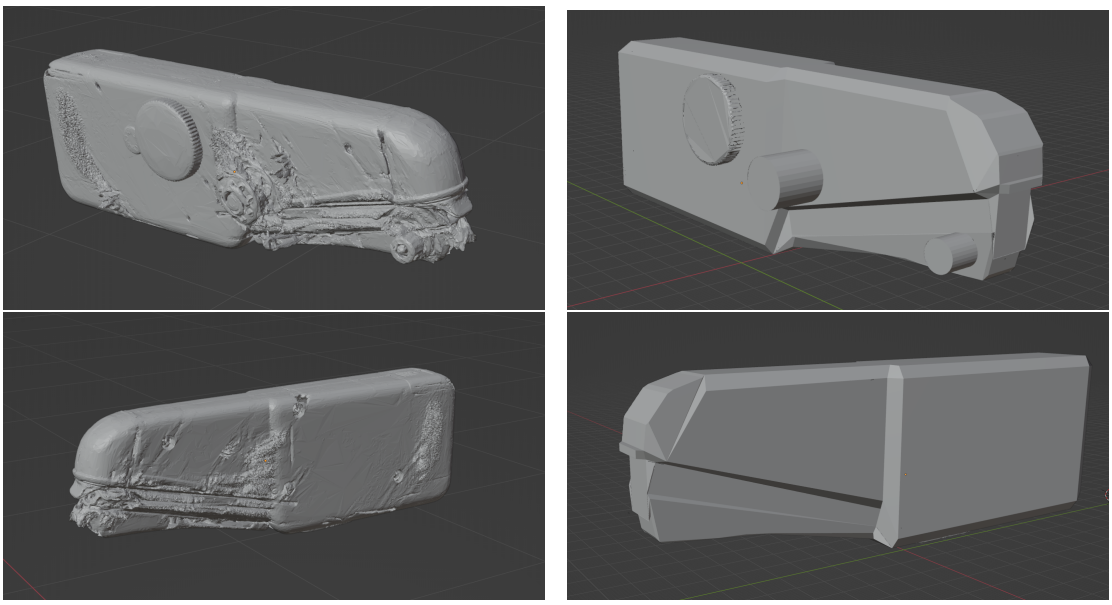


Figure 4.11: The images on the left side show the imported model from ImageVis3D. The images on the right show the final data set after being simplified and manually adjusted using blender.

The whole process of preparing the surface data for the prototype is done manually and time intensive. The time for preparation depends on the experience with the required programs of the person modifying the original data set as well as on the data set itself. If the modifier knows the steps which need to be executed and the CT scan has a good quality with few disturbances, the preparation process can be completed within one to three hours. However, if the data set is more complex, or disrupted as was the case in this thesis, the process takes multiple hours, if not days. Due to the lack of experience with the required programs and the inferior quality of the CT scan, the preparation of this surface data took multiple weeks.

4.3 Prototypical Implementation

The prototype consists of an Android and an AR application, which are responsible for the input and output for and by the user. Since it is a matter of two applications working together, the first subsection 4.3.1 deals with network communication. In the following subsection 4.3.2, the various input options are explained in more detail before the different states which can be assumed by the prototype are outlined in subsection 4.3.3. The implementation details conclude with an explanation of the different exploration functions in subsection 4.3.4.

4.3.1 Network Communication

The prototype implementation uses an Android application (client) deployed on the handheld tablet and an AR application (host) running simultaneously on the PC. In order for them to communicate with each other, the devices need to be connected to the same local network. Information can be exchanged using this network by sending different types of *NetworkMessage*s. The abstract class *NetworkMessage* holds a *NetworkOperationCode* which is used to identify the type of message and has been implemented by other classes to enrich the messages with message type specific information. The class diagram displayed in figure 4.12 shows the abstract class and all derived child classes.

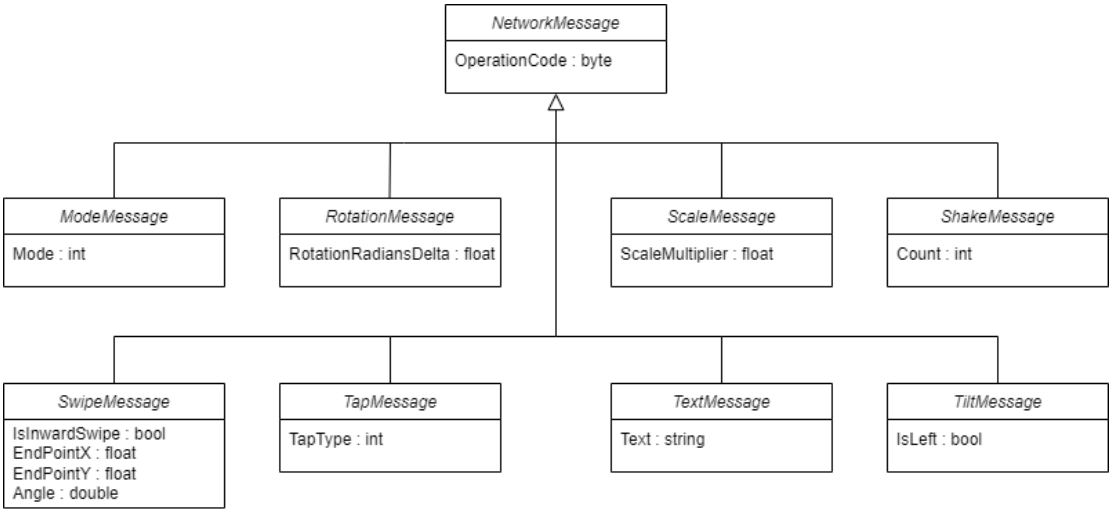


Figure 4.12: NetworkMessage and all derived message classes.

The information contained in the network messages needs to be serialisable, so only small data packages are sent. The different derived classes listed above can be described as followed:

ModeMessage Mode messages are used to communicate the mode change between client and host. The message holds an integer which is used to identify the *MenuMode* enum. The menu can either be *None*, *Selection*, *Selected*, *Mapping*, or *Analysis*.

RotationMessage Rotation messages hold the delta value of how far a twist has been executed. The rotation can then be directly applied to the object.

ScaleMessage Scale messages are used to communicate the scale multiplier which can be used to resize objects.

ShakeMessage Shake messages hold a count variable which tells the number of executed shakes using the HHD.

SwipeMessage Swipe messages contain the information of the swipe angle, the swipe direction (inward or outward), and the point where the swipe gesture ends.

TapMessage Tap messages are used to transfer which type of tap (single, double, hold start, hold end) has been executed.

TextMessage Text messages hold simple string values which are only used for debugging purposes.

TiltMessage Tilt messages are used to communicate in which direction, left or right, the HHD has been tilted.

The *Host* class and the *Client* class are responsible for receiving and sending these messages. They establish the connection via the local network and handle all message traffic and pass information to logical components.

Host

The *Host* class initialises the *NetworkTransport* in a method called in the Unity life cycle method *Start* and then adds a host before connecting. After setting up the *NetworkTransport*, clients can connect to the host and send messages which can be received using *NetworkTransport.Receive*. It is distinguished between three different kinds of events: *ConnectEvent*, *DisconnectEvent*, and *DataEvent*. In case of a *DataEvent*, the *BinaryFormatter* is used to deserialise the received information in form of a *NetworkMessage*. The message is then processed in the *OnData* method which handles it according to its *NetworkOperationCode*, calling any necessary actions. Apart from receiving, the host can also send messages to the client using the *SendClient* method, which serialises the messages with the BinaryFormatter and sends them with *NetworkTransport.Send*. Figure 4.13 shows a flowchart depicting the initialisation of the host application, as well as the rough process of communication with the client application.

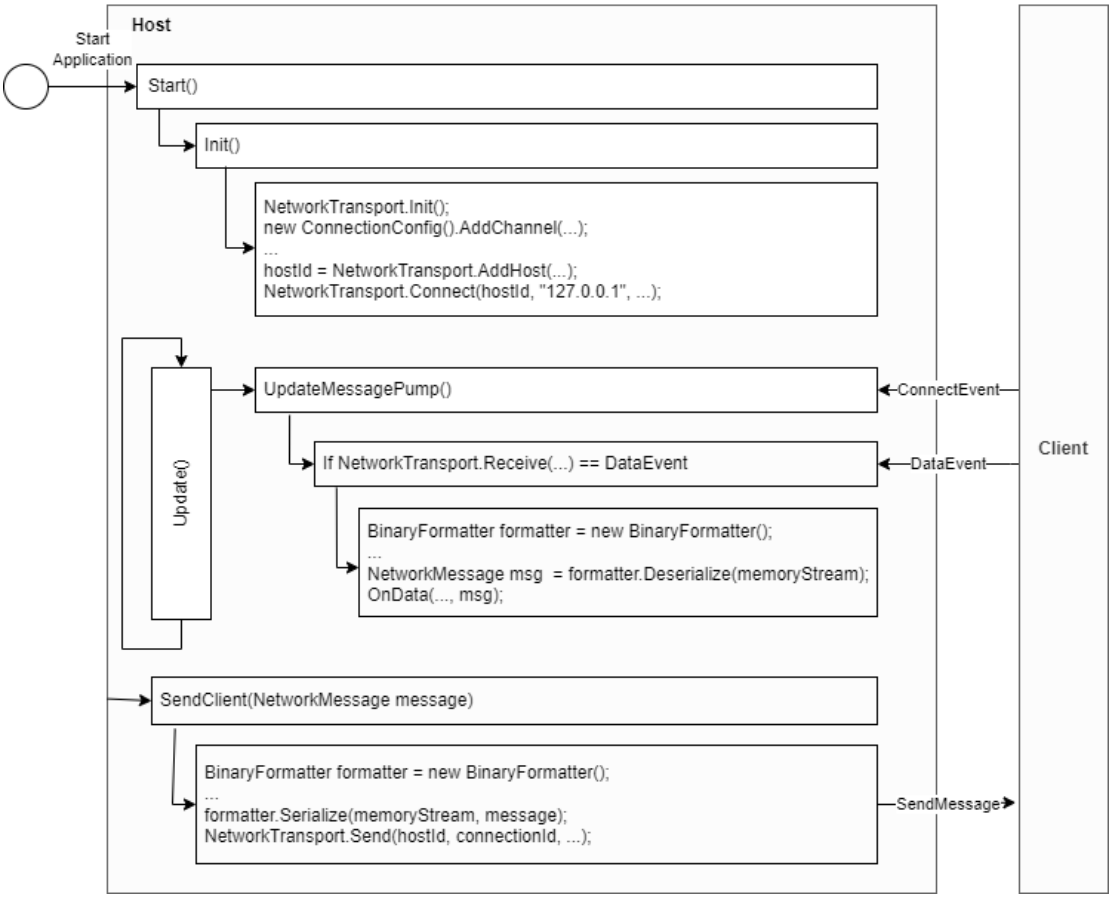


Figure 4.13: The process of initialisation within the host class, as well as the sending and receiving of network messages.

The host class transmits information to the client only after a reset or after a successful selection. All other mode changes can be detected by the client when sending input messages to the host, setting the new mode immediately.

Client

The implementation of the *Client*'s network communication is similar to the host's, which can be seen in figure 4.13. The main difference is that on connecting to the NetworkTransport, the client passes the IP address of the host device (PC) instead of the localhost IP address.

The *SendServer* method uses the *BinaryFormatter* to serialise network messages which are then sent using *NetworkTransport.Send* to the previously connected host id. Once messages are sent, they are checked for specific input which impact the client application, such as inward swipes and hold events. This allows the client class to set almost all prototype states through direct user input. The host only needs to inform the client about a mode change when shakes have triggered certain actions or when a selection was successfully executed.

The *UpdateMessagPump* method is called in the Unity life cycle method *Update* and used to check for received messages. Connection and disconnection events only trigger a debug log information while a data event allows the client to process received information. Again, the *BinaryFormatter* is used to deserialise received messages. The client receives only text messages which are used for debugging purposes and menu mode changes. The menu mode changes are needed to synchronise the host and client application.

4.3.2 User Input

User input can be executed in various forms utilising the handheld tablet. Actions regarding the tablet's spatial and touch-sensitive capabilities can be directly used by picking up movements and gestures. Indirectly, the position and rotation of the tablet is monitored using a tracking marker attached to its back. Figure 4.14 shows how the input recorded by the HHD is processed by the client application while the information gathered by the tracking sensor is passed to the AR application. Eventually, all input data is passed to the host class, which provides the information for the respective interaction classes. They are then responsible for applying the input to the model.

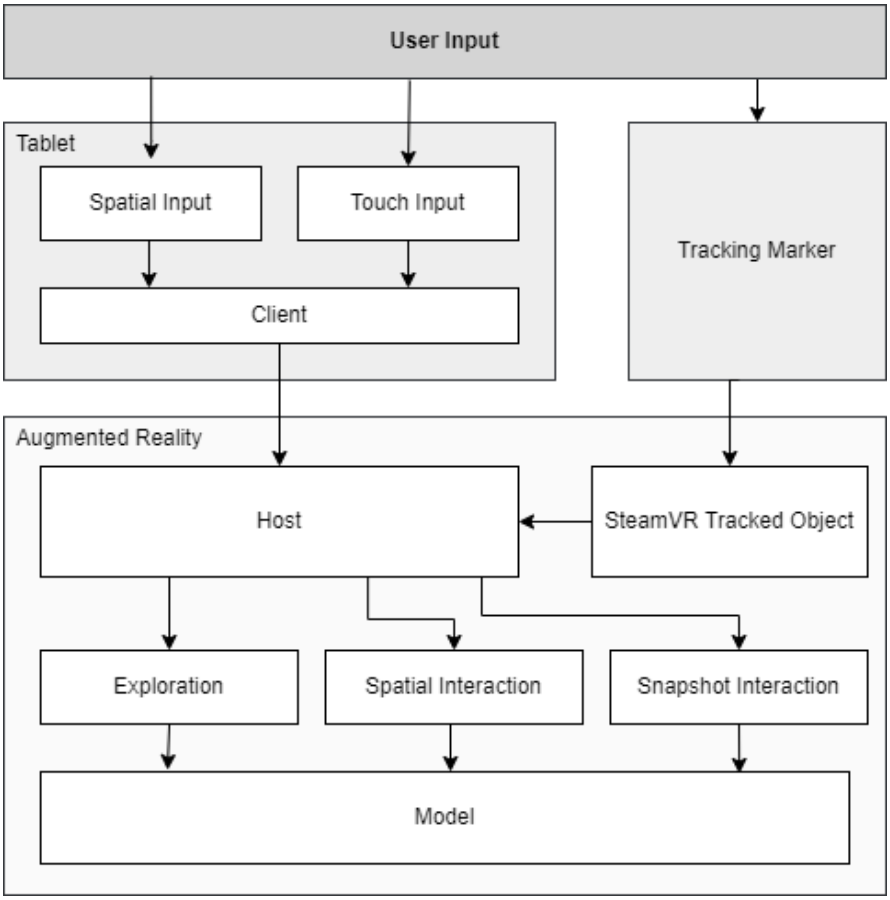


Figure 4.14: Interconnection between main components and main classes.

Spatial Input

The class *SpatialInput* is used to pick up and report actions such as the tilting and shaking the HHD. The *UnityEngine.Input* class is used to access the accelerometer and gyroscope data of the mobile device.

Shakes are detected using the *Input.acceleration.sqrMagnitude* property to compare the acceleration speed with a threshold. If the set time interval is valid, the shake counter is increased. A *ShakeMessage* is sent in case one or multiple shakes have been recorded.

Tilt movements are detected using the *Input.gyro* property. This Gyroscope instance holds the attitude of the HHD. When the device is tilted to the left or right in a horizontal position the x-attitude changes. If this value exceeds a previously set threshold, a *TiltMessage* is sent.

The accelerator and gyroscope may have varying sensitivities depending on the device. Therefore, if the client application is used on a different HHD, the thresholds for shaking and tilting may need to be adjusted.

Touch Input

Touch and swipe gestures are being tracked using the *TouchInput* class. This class uses the *FingersLite* repository[15] of Digital Ruby, which supplies multiple classes built to recognise specific finger based gestures. The *GestureRecognizer* class builds the basis for the recognition of such input. For each input type there is a specific class derived from the *GestureRecognizer* to track gestures. Following gesture recogniser can be differentiated:

TapGestureRecognizer is used to differentiate between single, double, and more tap gestures. They are differentiated using a *NumberOfTapsRequired* property which is compared to the counts of performed touches on the touch surface of the device.

SwipeGestureRecognizer is used to communicate executed swipes. The GestureRecognizer can be used to query the start and end point of the swipe which can be used to calculate its direction.

ScaleGestureRecognizer is used for callbacks about pinch gestures. The gesture holds information about the scale multiplier which can be passed on.

RotateGestureRecognizer recognises performed rotational gestures and holds the change in rotation radians.

LongPressGestureRecognizer informs about the begin and the end of a touch and hold gesture.

For each possible gesture a respective GestureRecognizer instance is created and callbacks are assigned. The recognisers are then added to a static *FingersScript.Instance* which invokes the callback when the corresponding gesture has been detected. Gestures can not be performed at the same time, apart from resizing and scaling which has been specifically defined to allow simultaneous execution.

[15] https://github.com/DigitalRuby/FingersGestures/blob/master/Assets/FingersLite/

Tracker Input

The position and rotation of the HHD is located using the tracking sensor mounted on its backside. This data can be received using the *SteamVR_TrackedObject* script which was added to a child gameobject of the *SteamVRObjects* gameobject, which in turn is a child of the *Player* prefab used to manifest the user in the AR Unity scene. The object tracking script allows to synchronise the movement of the tracking marker with the virtual gameobject in the augmented environment. It therefore plays a central roll for multiple interaction features. Most importantly, the data is used to position an overlay quad in the AR scene to render images on top of the tablet screen to make up for its insufficient resolution. Consequently, it is also used as a reference when aligning snapshots around the tablet overlay, but is also needed when positioning a snapshot as the initial position is relative to the position of the tablet. Positioned snapshots also always look towards the user, which is pinpointed with the position of the HHD. Additionally, the position and rotation of the tracker is important to find the intersection between tablet and virtual object when performing cuts, or when synchronising the movement of a selected object with the HHD.

4.3.3 Prototype States

As previously described in section 3.6 of the concept chapter, the user interface for the mobile application of the prototype is structured and built in a way that requires as little direct interaction (mode changes) as possible and consequently requires little eye contact with the device. Therefore, the prototype has only four states (main mode, selection mode, selected mode, exploration mode). The interconnection between these modes, represented by their UI images, is shown in form of a state machine in figure 4.15.

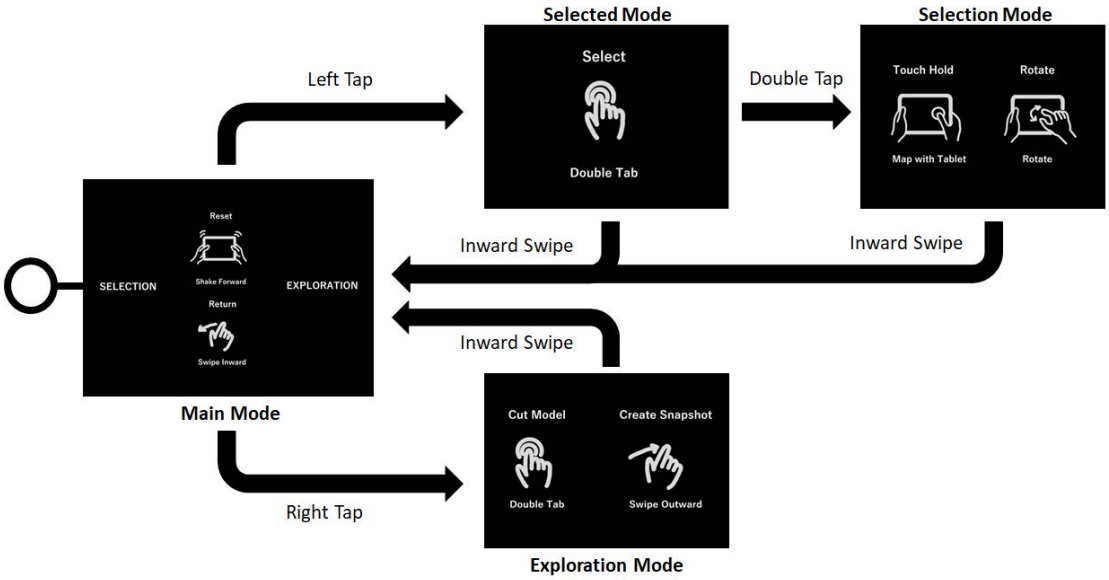


Figure 4.15: UI screens rendered in the different states of the prototype.

After a test run of the prototype with a new user showed that it is difficult for beginners to remember all functionalities, visual hints were added to be displayed on the tablet overlay. These should help the user to remember all available main functions of the currently active state. Icons are used to help with a quick glance, while a description is added in case the user has not internalised the different functions yet.

Main Mode

The application starts in the *main menu*, which offers two navigation options: The selection mode and the exploration mode. When in the main menu mode, the user can reset the augmented environment by removing snapshots with two shakes of the tablet. If no snapshots can be removed the model is reset to its original state, meaning frozen cutting planes are removed. These functions are illustrated in figure 4.16 which is rendered as tablet overlay in the main mode.

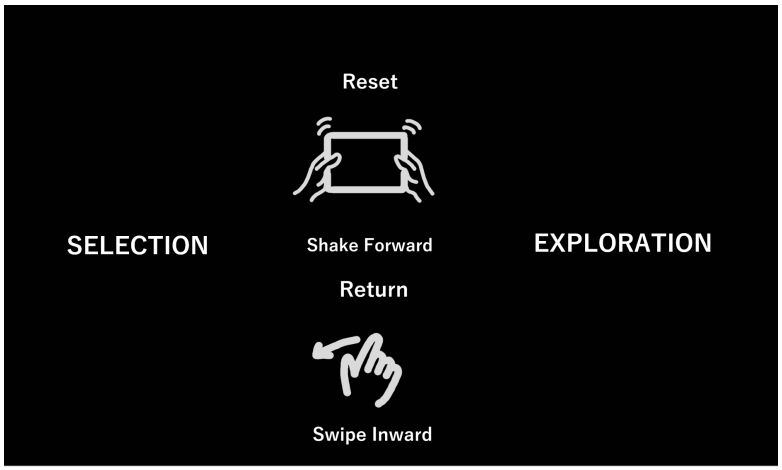


Figure 4.16: The main mode overlay depicts how a left tap starts the selection mode and a right tap starts the exploration mode. In addition, the reset function, which is executed using shakes, and the return functions are depicted.

Selection Mode

When the *selection mode* is started with a left tap, a ray is placed on top of the HHD. This serves as an extension of the tablet and is synchronised with its movements. The ray is equipped with the *Selector* script, which adjusts the colour of the ray in case of a collision. All objects with the *Selectable* script (model and snapshots) can be selected by a selector and react with a colour change when touched and selected (see figure 4.18). Selectable objects can also be frozen to prevent unwanted movement caused by the touch of a rigidbody. When an object is selected, the mode automatically switches to selected mode. If a swipe inward is detected the selection mode is cancelled and the main mode is restarted.

Selected Mode

The main function of the *selected mode* is to move and adjust objects in the user's environment. Touch gestures are used to resize, rotate, and reposition the selected object. In case the object in question is a snapshot, the snapshot image is displayed on the tablet's overlay surface. In addition, tilt movements can be used to calculate neighbouring slices and inspect the model layer by layer. During any interaction the user can perform a swipe inwards to exit the current mode and return to the main menu. Figure 4.17 shows the overlay UI shown in the respective state.

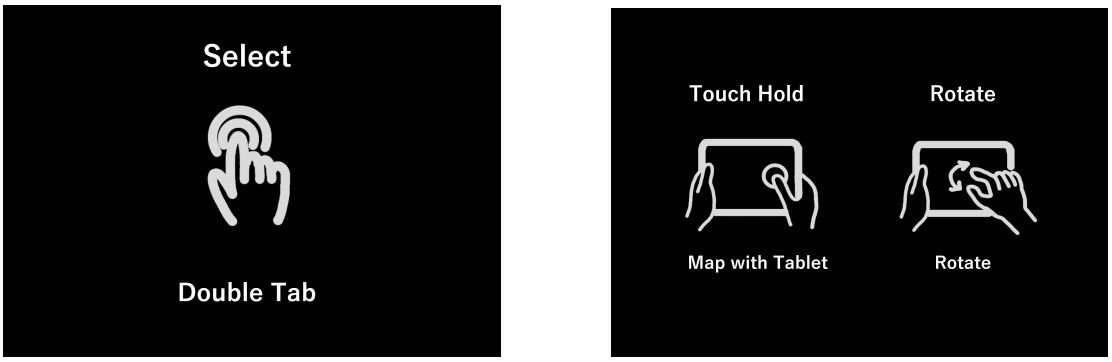


Figure 4.17: The left image shows the UI for the selection mode. During this mode the user can either double tap when the selection ray is intersecting an object or return to the main mode. The image on the right depicts some of the possible actions when the selected mode is active. Not all possible actions are presented to allow a better overview.

The following figure 4.18 is based on the state machine displayed in figure 3.4, and shows how these interactions affect the selected model or snapshot. The application starts in the main mode which allows the reset of the model and environment by shaking the HHD. If the user taps on the right side of the screen the selection mode is started. If the selection ray intersects a valid object and the user double taps, the selected mode is entered. When a snapshot was chosen, a shake removes it while a tilt starts the neighbour inspection. Otherwise, a pinch gesture is used to resize any object (model and snapshot), while a rotate gesture controls its orientation. A lasting long touch gesture synchronises the position and rotation between model and HHD.

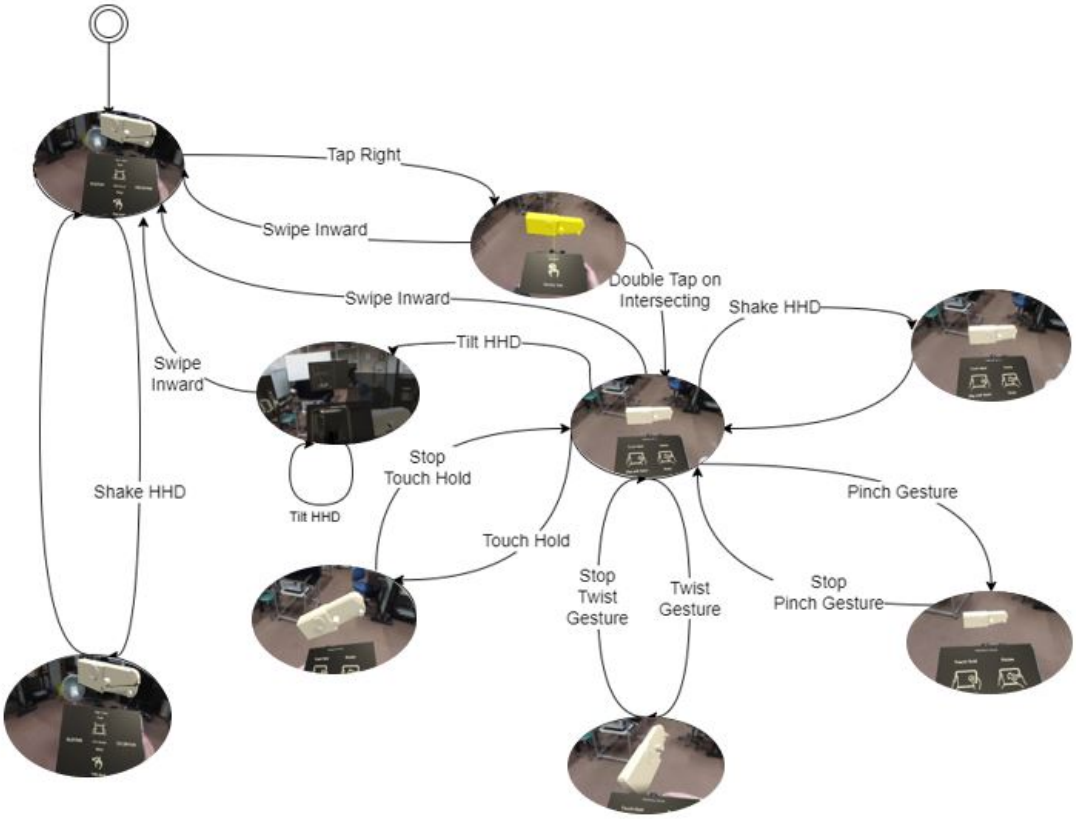


Figure 4.18: Interactions when the user has entered the selection mode.

Exploration Mode

The exploration is opened by tapping the right side of the touchscreen in the main menu. This mode focuses on the inspection of the internal structure of the three-dimensional model. A cutting plane is automatically attached to the tablet overlay, slightly in front of the tablet. This allows a part of the surface model to be visible between plane and overlay. The user then has the option to freeze the cutting plane in its current position. This triggers the calculation of the internal structure at the intersection using the original data set. The calculated image is then rendered as a texture on top of the cut. If the three-dimensional model should not be changed, the user can create a snapshot which is positioned within the augmented environment. These snapshots can also be aligned around the tablet to gather an overview. The main functionalities are depicted on the exploration specific overlay screen, which is displayed in figure 4.19.

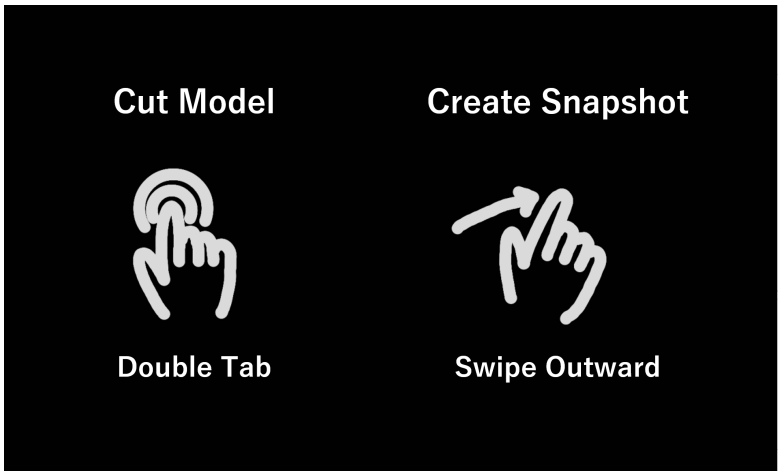


Figure 4.19: The UI for the exploration mode only depicts the two main functionalities: the cutting of the snapshot and the creating of a snapshot. The alignment of created snapshots has been left out for a better overview to focus on the main aspects of exploration.

How the described actions an be performed is shown by figure 4.20, which is based on the state machine displayed in figure 3.4. The application starts in the main menu which can switch to the exploration mode with a tap on the right side of the touch interface. In this mode the model can be temporarily cut by intersecting it with the HHD. A double tap then allows to freeze the cutting plane while an outward swipe creates a snapshot. The swipe direction is used to calculate the position in which the snapshot is positioned relative to the user. If snapshots are around, the user can perform a grab gesture to align all shots around the tablet, or if this is already the case, put them back to their original positions within the user's environment.

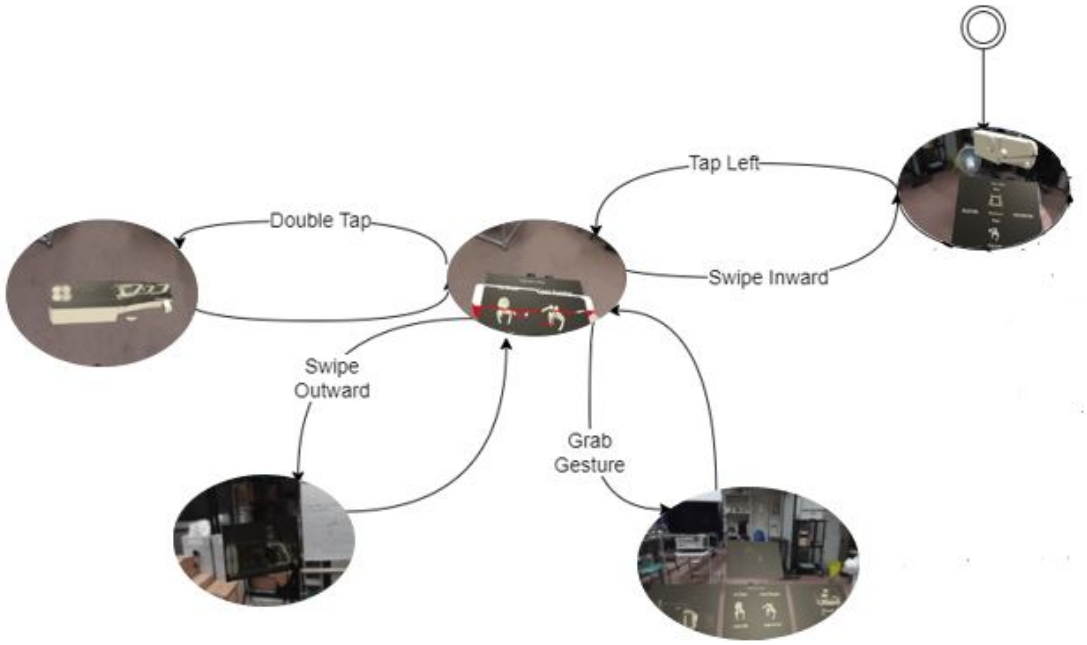


Figure 4.20: Interactions when the user has entered the exploration mode.

4.3.4 Data Exploration

The exploration features allow the user to inspect the stereoscopic dataset by querying monoscopic data. The two-dimensional data enhances the information presented by the surface data of the three-dimensional model. As the model rendered in Unity is a surface model, it does not contain any internal information. This allows for a higher rendering speed due to the smaller amount of data. Therefore, the internal structure must be calculated and fetched from the volumetric dataset whenever the user interacts with it. A sliceable object is a virtual object which can be explored by cutting off parts using the so-called slicer. The slicer is an object used to define the parts which should be pruned.

Temporary and Permanent Cutting

To avoid unnecessary computational costs, the prototype only calculates the intersection plane on explicit command of the user. For this reason, this prototype distinguishes between temporary and permanent cutting.

Temporary slicing is done in real time and describes the way the slicing plane removes parts of the 3D model when it is intersecting it. Upon removing the slicer from the object, the previously removed parts reappear. The *CrossSectionShader*[16] plugin[17] is used for

[16] https://github.com/Dandarawy/Unity3DCrossSectionShader
[17] https://assetstore.unity.com/packages/vfx/shaders/cross-section-66300

this effect. It was provided by Abdullah Aldandarawy but is now deprecated. It works by adding the *OnePlaneCuttingController* script to the object which is supposed to be cuttable, setting a slicer reference as its *plane* property. Both slicer and sliceable need to have the *CrossSectionShader* specific shader *OnePlaneBSP Shader* set as their shader. This way, if the slicer object intersects the slicable object, the shader stops rendering all parts which are located before the plane.

The cutting plane with this slicer script has been positioned in a short distance before the tablet overlay. This way a part of the surface model can be seen before the intersected part is removed (see figure 4.21). As the tablet otherwise covers the model, the displaying of the object's borders helps to understand the position of the slicer within the model. As the tablet is always positioned where the internal structure would otherwise be shown, no intersection needs to be calculated, which makes temporary slicing very fast and real time friendly.

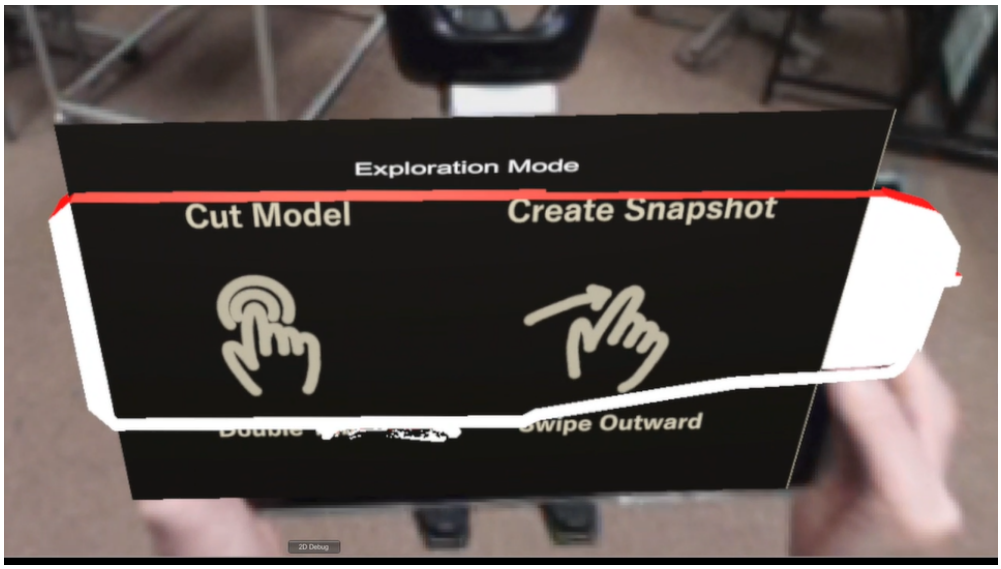


Figure 4.21: The temporary cutting plane is positioned slightly before the tablet overlay. This gives the user a visual hint of the surface's current position and providing better intuition of the tablet's position within the model.

In contrast to temporary cutting, the permanent type modifies the three-dimensional model permanently. The *EzySlice* project[18] is provided by LandVr and used to generate a new gameobject consiting only of the part of the 3D object not intersected by the cutting plane. For this, the *Slicer* script, which is attached to the plane, and the *SliceListener* script, attached to the model, are used. The SliceListener instance holds a reference of the slicer object and sets its *isTouched* property when detecting an intersection using the *OnTriggerEnter* method (see figure 4.22). When an intersection is detected and the user double taps, setting the *isTriggered* property, the process of cutting the slicable object is started.

[18] https://github.com/LandVr/SliceMeshes

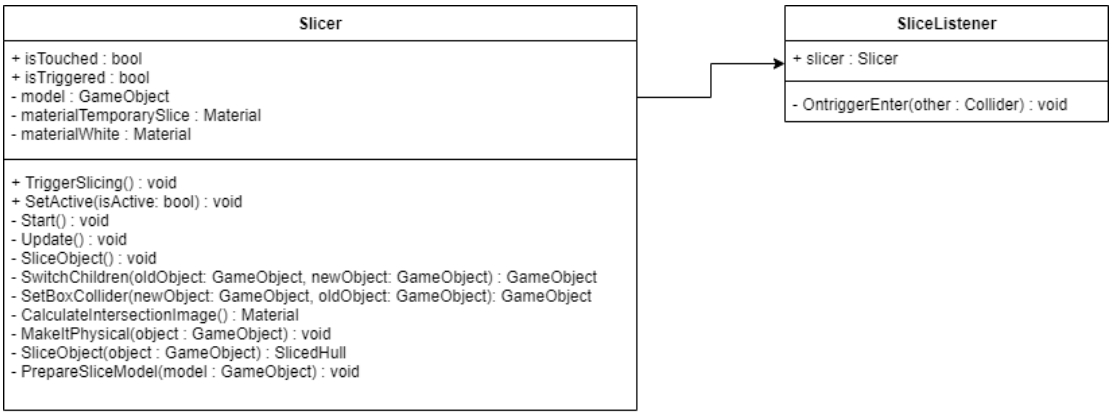


Figure 4.22: Class diagram of the Slicer and the SliceListener classes.

First, the intersection image is calculated (see subsection 4.3.4), playing a sound to inform the user when the calculation has started. For this audio feedback a camera shutter[19] was used, as the output will also be an image. The calculation of such an intersection is usually done within a few milliseconds. After the calculation, a *SlicedHull* instance is created from the slicable, and used to create a gameobject which does not contain the parts of the original gameobject which have been previously intersected. As the dimension of the model is changed by cutting away a part, its collider also changes. The collider box around the model needs to have the same dimension as the original model, to correctly map this position back to the volumetric data. As the origin planes of snapshots are saved in form of child elements, the trimmed object is then set as the parent gameobject to these planes, allowing for the old model to be destroyed. To prepare the model for additional slicing, the *Selectable* script, the *SliceListener* script, and the *OnePlaneCuttingController* script are set to enable temporary and permanent cutting. The *CreateUpperHull* method from the EzySlice project is used to create the new trimmed slicable, and allows to set the material of the face which is set in the position where the cut was performed. Figure 4.23 shows how the new object has set the calculated intersection image at the position of the cut.

[19] https://freesound.org/people/kwahmah$_0$2/*sounds*/260138/

Figure 4.23: Example of a permanently changed model using the permanent cutting functionality of the prototype. The intersected part has been removed and a material containing the calculated volumetric data texture has been added to the intersecting face.

Intersection Calculation

The calculation of the intersection plane and the resulting image is the point where the prototype makes the connection between surface model data and volumetric data. This calculation logic utilises multiple classes to find the intersection between plane and model and calculate the information at this position using the data of image slices.

The *Model* class, shown in figure 4.24, reads all exported image slices from the volumetric data and holds them in form of a bitmap array. In addition, the class also has properties for the dimensions of the data set (*xCount, yCount, zCount*). These properties and the data array are automatically initialised when a class instance is created. The model class is used to get the intersection of a plane with the model and also to query the intersection image from another class.

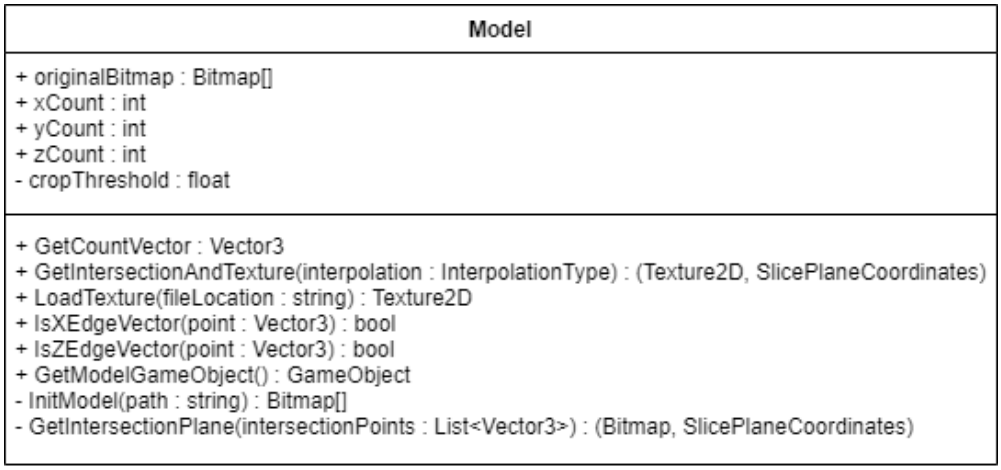


Figure 4.24: Model class with all properties and methods.

To calculate a sectional image, the position of the intersection plane in relation to the volumetric data set must first be determined. The model method *GetIntersectionAndTexture* creates a *ModelIntersection* instance, which is a class used to calculate the position of the plane within the model. This is achieved by positioning one object at each corner of the cutting plane and moving it towards its centre until it touches the collider of the model as can be seen in figure 4.25.

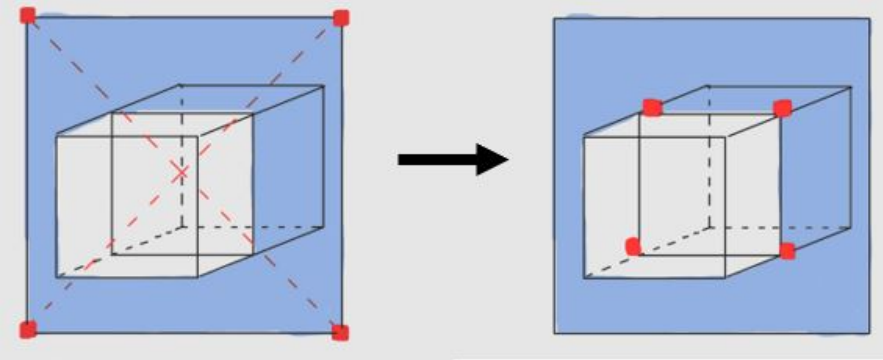


Figure 4.25: The section plane is shown in blue, while the points used to determine the section boundary with the 3D object are shown in red. The left image depicts how the points start at the edges of the plane and move towards its centre. In the right image the points have stopped on colliding with the boundary of the slicable object.

It is therefore important to intersect the model through the centre of the plane, otherwise the collider may not be found. As the contact points are set to be children of the sliceable object, their positions can be normalised by adding half of the size of the selectable to their position. The relative position within the volumetric dataset can be calculated using the algorithm 4.1.

Algorithm 4.1: Calculation of the intersection points within the model data

1: $model \leftarrow$ model instance holding the dimensions of the volumetric data
2: $size \leftarrow$ size of the model collider
3: $normalisedContacts \leftarrow$ touchpoints of the plane edges with the model collider
4: $positionsWithinModel \leftarrow$ new List to return
5: **for** var $contactinnormalisedContacts$ **do**
6: $\quad relativePosition.x \leftarrow (contact.z/size.z) * model.xCount$
7: $\quad relativePosition.y \leftarrow (contact.y/size.y) * model.yCount$
8: $\quad relativePosition.z \leftarrow (contact.x/size.x) * model.zCount$
9: $\quad positionsWithinModel \leftarrow relativePosition$

As can be seen in this algorithm, the x and z values are switched. Unity uses the left handed coordinate system, while myVgl works with the right handed coordinate system. As the surface data is computed using Unity, while the volumetric data for the internal structure has been exported from myVgl, the x- and z-axis need to be changed to avoid errors.

The relative contact points are then passed on to the *GetIntersectonPlane* method which returns the final intersection image together with the coordinates of the plane. First, an instance of *SlicePlane* is created, which is used to establish the link between the position of the plane within the model and the model itself. To do this, the class *PlaneFormula* is set to calculate the equation of the plane using the algorithm shown in algorithm 4.2. Even though the retrieval of the intersection points provides four coordinates, only three are required and used to calculate the plane formula.

Algorithm 4.2: Calculation of the plane

1: $point1 \leftarrow$ first of the intersection points
2: $point2 \leftarrow$ second of the intersection points
3: $point3 \leftarrow$ third of the intersection points
4: $a1 \leftarrow point2.x - point1.x$
5: $b1 \leftarrow point2.y - point1.y$
6: $c1 \leftarrow point2.z - point1.z$
7: $a2 \leftarrow point3.x - point1.x$
8: $b2 \leftarrow point3.y - point1.y$
9: $c2 \leftarrow point3.z - point1.z$
10: $a \leftarrow b1 * c2 - b2 * c1$
11: $b \leftarrow a2 * c1 - a1 * c2$
12: $c \leftarrow a1 * b2 - b1 * a2$
13: $d \leftarrow (-a * point1.x - b * point1.y - c * point1.z)$

The result values $(a,\ b,\ c,\ d)$ are then be inserted into the plane equation 4.1

$$ax + by + cz + d = 0 \tag{4.1}$$

to check if a point lies on the plane.

After setting up the plane formula, the edge points of the cutting plane can be calculated within the volume data. For this, the x, y, and z values of each possible edge point are inserted into the formula. If the result is 0, the point lies on the plane, otherwise it does not. Once the edge points of the intersection have been determined, the width and height of the cutting image can be derived and this information can be used to determine the step size between the pixels.

Once the SlicePlane instance has been created, which also validates, that the intersection can be done, the model instance calls *SlicePlane.CalculateIntersectionPlane* which is described in algorithm 4.3.

The algorithm iterates through the bitmap with the size of the new image and calculates the value for each pixel. The image values can be calculated using either the nearest neighbour interpolation, bi-linear interpolation, or none. Nearest neighbour rounds the x and y position of the value within the original image which leads to inferior quality of the result image and visible edges as neighbouring values are not taken into the calculation. Bi-linear on the other hand takes adjacent pixel values into consideration, interpolating them and utilising a delta value. While it achieves a better image quality than the nearest neighbour interpolation, it also requires more calculation and is therefore slower. During the development of the prototype, the nearest neighbour

Algorithm 4.3: Calculation of the intersection image

```
1: plane ← holds all the information about the plane
2: model ← holds the model pixels in form of a bitmap array and its dimensions
3: function CalculateIntersectionPlane(alternativeStartPoint)
4:     resultImage ← newBitmap(plane.Width, plane.Height)
5:     startPoint ← alternativeStartPointifnotempty, welseplane.StartPoint
6:     currVector1, currVector2 ← startPoint
7:     for w ← 0, 1, ..., plane.Width do
8:         currVector1 ← startPoint + w ∗ plane.XSteps
9:         for h ← 0, 1, ..., plane.Height do
10:            currVector2 ← currVector1 + h ∗ plane.YSteps
11:            currOriginalBitmap ← model.originalBitmap[currVector2.x]
12:            result ← value of currOriginalBitmap using currVector2[z, y] for interpolation
13:            resultImage set resultvalue at position w, h
14:    return resultImage
```

interpolation could calculate the intersection within a few milliseconds, while the bilinear method was more noticeable and took sometimes up to a few seconds. As the computation time is important for this prototype, the nearest neighbour interpolation was set to be the default method.

The described method *CalculateIntersectionPlane* accepts an alternative start point which is used for calculating neighbour slices of snapshots (see subsubsection 4.3.4). This way the same method can be reused, as the plane stays the same and only moves a step along the main axis.

After the calculation has finshed, the method returns the intersection image which is passed on by the model method *GetIntersectionPlane* along with the plane information. There, the image is saved to the file system using a bmp and a png format. The png file is then loaded again in form of a Unity *Texture2D* which can be added to any material as a main texture.

Snapshots

Snapshots are momentary recordings of the tablet position within the three-dimensional dataset. They visualise the internal structure at the position of their capture while also saving the position of their origin.

The *Snapshot* class holds information for its positioning, rotation, origin, and the plane information for the calculation of its intersection image. The properties *Viewer* and *IsLooking* are used to allow the object to always be visible to the user. The viewer property holds the position of the HHD, so the *LookAt* method can be used to make it always face the user. As a snapshot can be pulled towards the tablet and rendered parallel to the screen, the position in the user's environment needs to be saved to allow for the object to be returned when they are no longer needed around the HHD. When a snapshot is created the cutting plane is copied and saved as an inactive child object of the slicable model. The snapshot holds a reference of this copied plane, which is

activated whenever the user inspects the object. Along with the intersection image, the shot also saves the PlaneCoordinate instance to be able to later calculate neighbouring slices without having to recalculate the intersection part.

The *SnapshotInteraction* class is used to handle all interaction requests concerning snapshots. When the host receives a swipe outward message, it is used to call the *HandleSnapshotCreation* method. The swipe angle is passed along to determine the position of the snapshot relative to the position of the user. The calculation of the position using the angle of the swipe direction is described in algorithm 4.4.

Algorithm 4.4: Calculation of the snapshot position

1: $hhdPosition \leftarrow$ position of the HHD
2: $centeringRotation \leftarrow -90$
3: $distance \leftarrow$ configurated value of distance between HHD and snapshot
4: $snapshotPosition \leftarrow hhdPosition + Quaternion.AngleAxis(swipeAngle +$ y rotation of HHD $+centeringRotation, Vector3.up) * Vector3.back * distance$

The position and rotation of the HHD can be read using the tracking marker, while the distance between HHD and snapshot can be set in the configurations. The swipe angle is received on notification of a swipe event from the client and uses the *AngleAxis* method to calculate the position relative to the user. The height of the HHD is adapted while the x and z values depend on the calculation.

When a snapshot is created, the intersection image is loaded and set as its texture. In addition, the current position of the HHD is saved to the *OriginPlane* property, while a plane is created in its position. This origin plane is always deactivated and only gets set to visible when the snapshot is selected.

To allow a fast alignment of all snapshots a static approach has been chosen. Five deactivated planes are positioned around the tablet overlay. Their position and scale is used to place the snapshots in their place as can be seen in figure 4.27. In addition, the tablet overlay is set as their parent so that all movements and rotations are automatically applied to them. To allow the repositioning in their original place, the position and scale is updated in the snapshot instance before aligning them. They can be used to undo the alignment if needed, as seen in figure 4.26.

Figure 4.26: All snapshots are positioned in the user's environment.

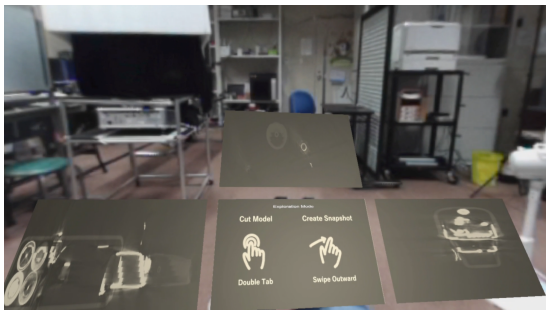


Figure 4.27: All snapshots are aligned around the HHD.

The neighbour inspection is a layer by layer approach which allows the user to inspect how the intersection image would look if the snapshot was places slightly deeper or not as deep in the object compared to its actual position. The original plane coordinates of the selected snapshot are used to create a SlicePlane to call the *CalculateNeighbourIntersectionPlane* method and pass the direction of the inspection. The direction is given by the side of the tilt recorded by the host. A leftward tilt sets the snapshot deeper into the model, while a rightward tilt moves the plane backwards into the intersected area. The axis along which the movement is done is calculated using the start point. It is controlled if it is an edge value of either the x-, y-, or z-axis. The start point is then shifted along the movement axis and handed as an alternative start point to the *CalculateIntersectionPlane* method which handles the creation of the intersection image. The result image and the plane coordinates with the new start point are then handed back to the SnapshotInteraction. A temporary snapshot is created and displayed on the tablet overlay, with its origin plane, which is now shifted. The shift performed with the starting point within the volumetric data now needs to be translated to the AR environment as described in algorithm 4.5.

Algorithm 4.5: Calculation of the new origin plane position

1: $direction \leftarrow originalStartPoint - newStartPoint$
2: $dimensionKey \leftarrow surfaceModelDimensions/volumetricModelDimensions$
3: $offset \leftarrow dimensionKey * direction * modelScale$
4: $newPosition \leftarrow originalPlanePosition + offset$

The conversion of the step size from the volumetric dimensions to the surface dimensions is done by first getting the size of the jump between the slices, subtracting the new start point from the original one. A dimension key is calculated by dividing the dimensions of the data set in AR by the dimensions of the data set of the volumetric data. The offset can then be calculated by multiplying this conversion key with the step size along with the scale factor. The scale factor is important as the model could have been resized.

The new position of the neighbour's slice origin plane is then set using the sum of the previous origin plane and the calculated offset. When the neighbour inspection is cancelled using a swipe in gesture, the temporary snapshots and their origin plane which are again children of the model, are removed.

Figure 4.28 shows how the temporary snapshot image is displayed on the tablet overlay at the bottom of the image. The origin plane is rendered intersecting the model on the right side. As the neighbouring slice is displayed on the tablet, it can be held up to compare it to the original slice, which is positioned in the central background of the following figure.

Figure 4.28: The neighbour slice is shown on the tablet overlay instead of the selected snapshot which allows a direct comparison to the original snapshot.

4.4 Limitations

In the course of the design and implementation of the prototype a number of limitations were encountered.

4.4.1 Hardware

The development of the prototype was finalised using the hardware described in section 4.1. This setup had a varying performance between 45 and 90 *frames per second* (FPS). In addition the resolution of the used HMD did not suffice to be able to read and recognise text and images displayed by the tablet. This limitation could be worked around using a more high resolution HMD, such as the Varjo XR-3[20] which offers 1920 pixel per eye for the focus area.

4.4.2 Network

When working with two connected applications, the main focus lies on the network traffic. As the client connects to the host devices using its IP address, both devices need to be connected to the same network. In addition, the host device must also deactivate its firewall, as a connection is otherwise not possible.

4.4.3 Touch based input

In contrast to the design mentioned in the concept, it was not possible to implement touch gestures with more fingers than required. Therefore, the grab gesture can only be

[20] https://varjo.com/products/xr-3/

performed with two fingers and is strictly speaking a pinch gesture. Also, the rotational gesture can only be performed using the necessary two fingers and does not work when more fingers touch the screen.

4.4.4 Frozen cutting plane

When using the shader plugin *OnePlaneBSP*, cutting planes can be dispatched and therefore appear as frozen. As the prototype enables the user to place multiple cuts, other planes can be positioned. Due to the shader attached to the planes, previously removed parts could reappear. To avoid this, parts needed to be removed permanently, changing the object. Therefore, frozen cutting planes cannot be edited, the whole object would need to be reset. The shader plugin is now used for temporary slicing, as it requires only one slicing plane.

4.4.5 Intersection orientation

Calculating a cut within the volumetric data does not allow the user to cut across edges only. The plane needs to be positioned in a way, straight or angled, that cuts the model in its whole height, length, or width. Figure 4.29 visualises how only full cuts can be made.

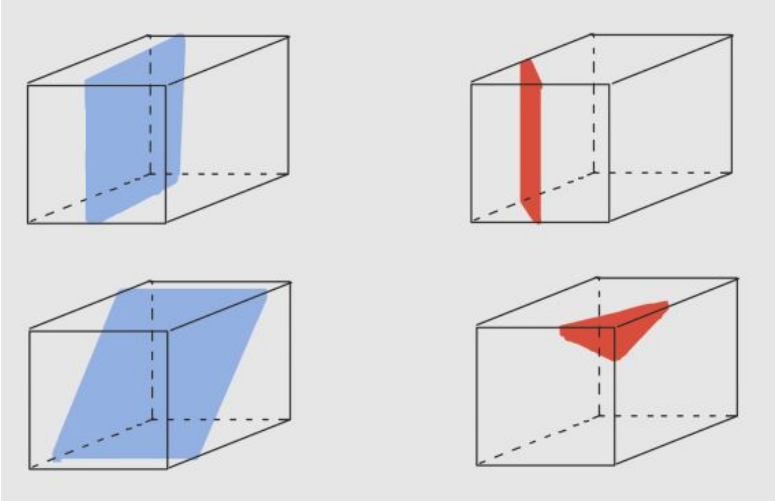


Figure 4.29: The blue intersections described on the left side show valid positions for the cutting plane. The plane intersects the object completely. The red intersections seen on the right are invalid because the positions never completely cut an entire axis.

4.4.6 Cutting plane position within model

The position of an object within another object is hard to find within Unity. Even when using collision points, the position within the object is not exact and often wrong. As Unity offers no such functionality, a workaround was used. A child object was positioned on each edge of the cutting plane and moved towards the centre of the plane until they collided with the intersecting entity. This way the outside of the collider could be found.

Because of this workaround, the cutting plane has to be positioned in a way, that the model lies in its centre. Otherwise the moving edge points might miss the collider box of the model.

4.4.7 Calculation format

The slices of the volumetric dataset were stored in a bitmap. The bitmap is not native to Unity, but is provided by *System.Drawing* and had to be imported using an rsp file. The bitmap was then also used to save the calculated cutting planes, but could not be converted into a Unity texture because converting a bitmap causes Unity to crash. Therefore, the file had to be saved in the file system and loaded separately. Since the *LoadImage* method of *Texture2D* only supports certain types, the byte files loaded from a bitmap file could not be displayed. Therefore, the calculated intersection was saved in the form of both, a bitmap and a png file, with the png file being loaded to use as a texture for the intersection.

Chapter 5

Evaluation

The aim of this master thesis is to find out how to enable the user to intuitively interact with and explore three-dimensional data using a *handheld device* (HHD). A user study is conducted to evaluate the prototype, find its weaknesses and strengths, and discover possibilities of improvement. The first part of this section (subsection 5.1) describes how the user study was conducted, gives an overview of the participants and evaluation structure, before going into detail about the results. The section concludes with subsection 5.2, discussing the findings and possible improvements.

5.1 User Study

The user study was carried out to determine which interactions and functions of the prototype and the concept are easy, intuitive, or difficult to use and understand. In addition, it was to be determined to what extent a tablet can be used as an input device in an augmented environment and which functionalities are perceived as supportive when interacting with 3D objects.

5.1.1 Design

The design of the study was created in a way to allow the user to understand the basic principles and functions of the prototype before interacting with it. It was divided into three parts: The introduction, where the idea of the concept was explained and basic information about the subject was collected. The interaction part, consisting of a short tutorial followed by letting the user interact with the prototype. Finally, a questionnaire and an interview were used to collect the users' impressions and opinions about the concept and the application.

Introduction

The study started with the users being made aware of their data protection rights and were asked to sign a respective paper (see appendix B). The study was conducted in Japan where the *Act on the Protection of Personal Information* (APPI)[1] is used to

[1] https://www.ppc.go.jp/en/

protect the data of private individuals. Similar to the APPI, the General Data Protection Regulation (GDPR)[2] is used in the European Union to protect the private data of individuals. Since 2019, there is a standing adequacy decision between Japan and the European Union [3], meaning they recognise each other's data protection laws as providing adequate protection for personal data. On the basis of these mutual respect for each others laws, the participants of this study were informed about their rights under the GDPR.

After the data protection formalities, the participants were first introduced to the general concept of the prototype and asked to fill in a small questionnaire about their person, followed by questions about their previous experiences with Mixed Reality, 3D Models, and touch-sensitive handheld devices.

Prototype Usage

The introduction was followed by the main part of the study, in which the test persons could familiarise themselves with the prototype and use it. This phase lasted between 30 and 50 minutes. The users were introduced to the concept and features of the prototype in a small tutorial. First, the structure of the menu was described before short videos on how to use the prototype were played. Questions from the participants were answered immediately and the videos could be played as often as desired. This part lasted between 15 to 20 minutes, depending on how fast the presented functionalities were understood.

After this basic walk through the participants could use the prototype in the course of guided instructions. The users were asked to perform the following tasks while being guided by an instructor which gave comments and support along the execution of the asked commands:

- **Task 1:**
  - Rearrange the model in the augmented environment by adjusting its position, rotation, and size.
- **Task 2:**
  - Create snapshots and position them in the augmented environment.
  - Align all existing snapshots around the tablet and compare them. Put them back at their original position.
  - Select a snapshot as a starting point and investigate the model layer by layer browsing through its neighbour slices.
- **Task 3:**
  - Freeze the cutting plane at a specific point within the three-dimensional model, effectively cutting it.
- **Task 4:**
  - Remove all taken snapshots.
  - Reset the model.

[2] https://gdpr.eu

[3] https://ec.europa.eu/commission/presscorner/detail/en/IP$_1$9$_4$21

The instructions covered all available functionalities and took the test subjects between 10 to 25 minutes. After the participants had completed this part, they were free to explore the prototype further on their own. Assistance was provided when users forgot a command or asked for help. To give them another task, all participants were asked to find the batteries in the model. The test subjects could explore the prototype as long as they chose, which was between 5 to 8 minutes in average.

Questionnaire and Interview

After using the prototype, participants were asked to fill in a questionnaire and take part in an interview. The *User Experience Questionnaire* (UEQ)[4] was used to gain a comprehensive impression of the user experience. It uses 26 pairs of attributes describing the product to measure its usability (attractiveness, dependability, efficiency, perspicuity) and user experience aspects (novelty, stimulation). Each item of the questionnaire consists of a pair of aspects, the positive aspect and the opposite, and a seven-level gradation in between as displayed in figure 5.1.

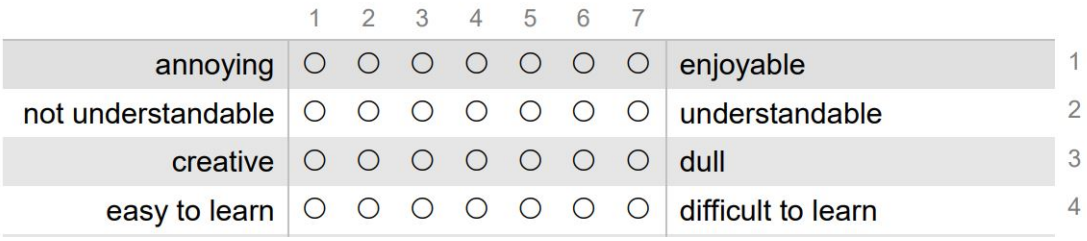


Figure 5.1: Example of a pair of aspects and the seven-level gradation which the test subjects can choose from.

In order to get a more detailed insight into the participants' feelings about the use of the prototype and the interaction possibilities, the study was completed with a semi-structured interview. This interview form allows to divert from the guiding questions (see appendix B) and adjust the direction of the interview depending on the answers of the participant. It allowed the interviewer to refer to observations made during the prototype usage part while always being able to go back to the main line of questioning. These questions address the general usage of the prototype, including the handling of the HHD, and specifics about the usage of different features.

5.1.2 Environment

The study was executed in a computer laboratory at the *Iwate Prefectural University* (IPU)[5]. It offered an open space of 2.5m x 2.3m, which gave the participants enough space to move around and interact with the prototype (see figure 5.2).

[4] https://www.ueq-online.org/
[5] https://www.iwate-pu.ac.jp/en/

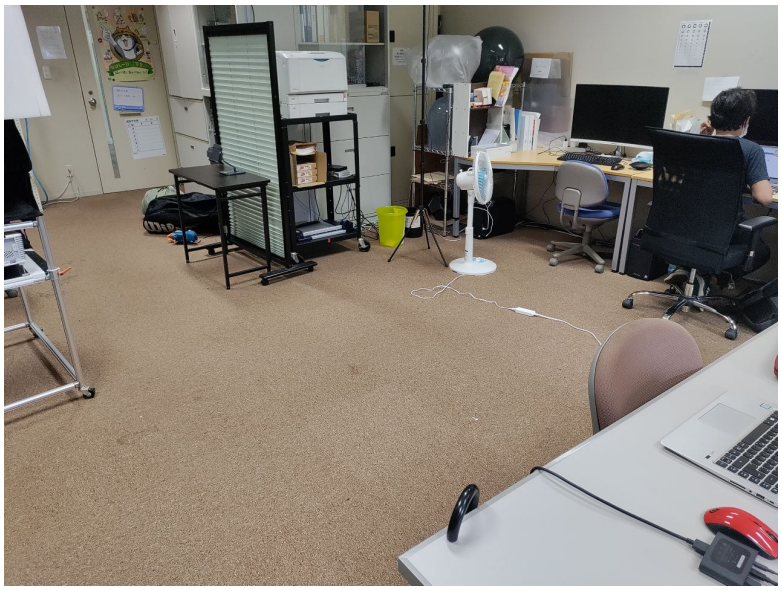


Figure 5.2: Real world environment in which the study was conducted.

5.1.3 Participants

This preliminary study focuses on the design evaluation, therefore a qualitative study investigating the usage of the prototype using a test population holding five persons was deemed sufficient to make qualitative measurements. The test population consisted of four men and one woman, aged between 23 and 32 years. Three out of the five participants needed eye correction and wore their prescribed glasses during the execution of the study. Four test persons were students, one was a research assistant. Every participant had previous experience handling touch-sensitive tablets, three had used 3D modelling for multiple years, but only two persons had thorough experience of using a *Mixed Reality Environment* (MRE).

5.1.4 UEQ Results

The data analysis tool provided by UEQ was used to evaluate the input generated by the users. As it uses a seven point grading system the grading ranges from -3 meaning horribly bad to +3 meaning extremely good. The following results use these grades to rank the different items and aspects. The mean is calculated leading to the result being over +2 or below -2 to be unlikely due to the different opinions and grading tendencies of the individuals. Every mean over 1.5 is seen as good.

The results of the questionnaire were summarised into six main scales (attractiveness, dependability, efficiency, novelty, and perspicuity). The diagram in figure 5.3 which visualises the mean rating per scale, shows that the prototype was rated positively in every aspect. The grey bars illustrate the mean rating of the scales, which are all in the positive range above 0. The median (and mean) of five out of six scales lie above 1.5 which can be interpreted as a very good result.

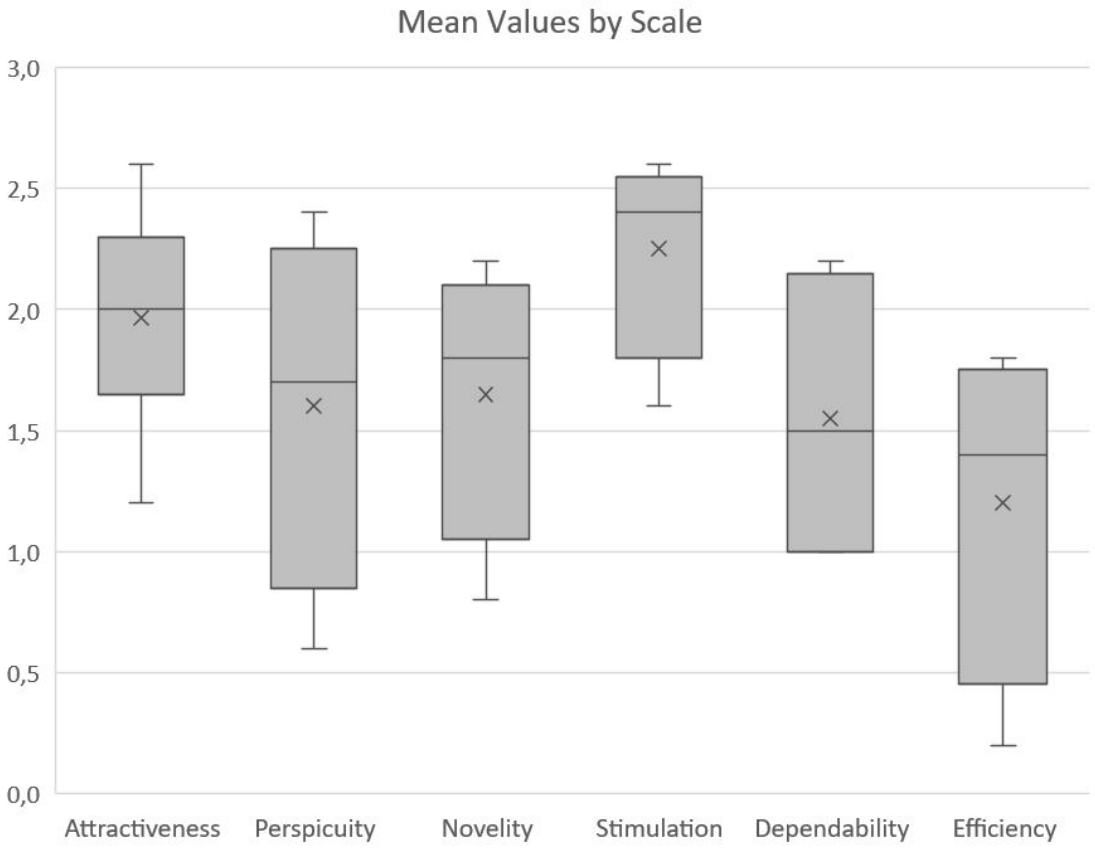


Figure 5.3: Diagram cumulating the mean of all ratings grouped the main scales.

The table displayed in figure 5.4 shows how the aspect *efficiency* has the highest answer variance (1.86), directly followed by perspicuity (1.46). The stimulation (0.38) and novelty (0.39) aspects on the other hand, have the least rating variance.

Scale	Mean	Median	Variance	Std. Dev.
Attractiveness	1,97	2,00	0,66	0,81
Perspicuity	1,60	1,70	1,46	1,21
Efficiency	1,20	1,40	1,86	1,36
Dependability	1,55	1,50	0,92	0,96
Stimulation	2,25	2,40	0,38	0,61
Novelty	1,65	1,80	0,39	0,63

Figure 5.4: Mean, median, variance, and standard deviation of the ratings by scale.

If the scaling factors are broken down to the 26 individual items, there are only 3 items in total which have a mean value below 1. The *speed* of the prototype has the worst mean rating with 0.2, followed by it being described as *complicated* with 0.6. It also was only described as *leading edge* with a mean of 0.8. Extremely positive items were the attributes *good* (2.6), *interesting* (2.6), *understandable* (2.4), and *motivating* (2.4).

Answer Distribution

Figure 5.5 shows the distribution of the responses of the study participants in relation to the attribute pairs. Test subjects were able to rate the use of the prototype by choosing a gradation between a pair of attributes (e.g. conservative and innovative). The attribute pair consists of two opposing items, one of which is a positive description of the prototype, the other is not. The attributes on the left side are used for the negative description and all ratings tending towards them are coloured in a reddish tone (1-3). The attributes on the right side of the diagram are used for a positive description and ratings which tend towards them are shown in a greenish tone (5-7). The grey ratings show those ratings which tend towards neither attribute.

The figure shows, that all ratings lean into a positive direction. However, some answers are somewhat divergent. Two items, *unusual and leading edge*, and *slow and fast*, clearly stood out as they each had one very negative rating and otherwise only neutral and positive ratings. The attribute pair *slow and fast* for example was rated by one user as completely slow (1), by one as completely fast (7), by one as rather fast (5) and the remaining two participants felt it was neither (3).

The difference in opinion regarding the rating of the *slow and fast item* which is depicted in the diagram explains why the efficiency scale has the highest overall variance compared to other scales.

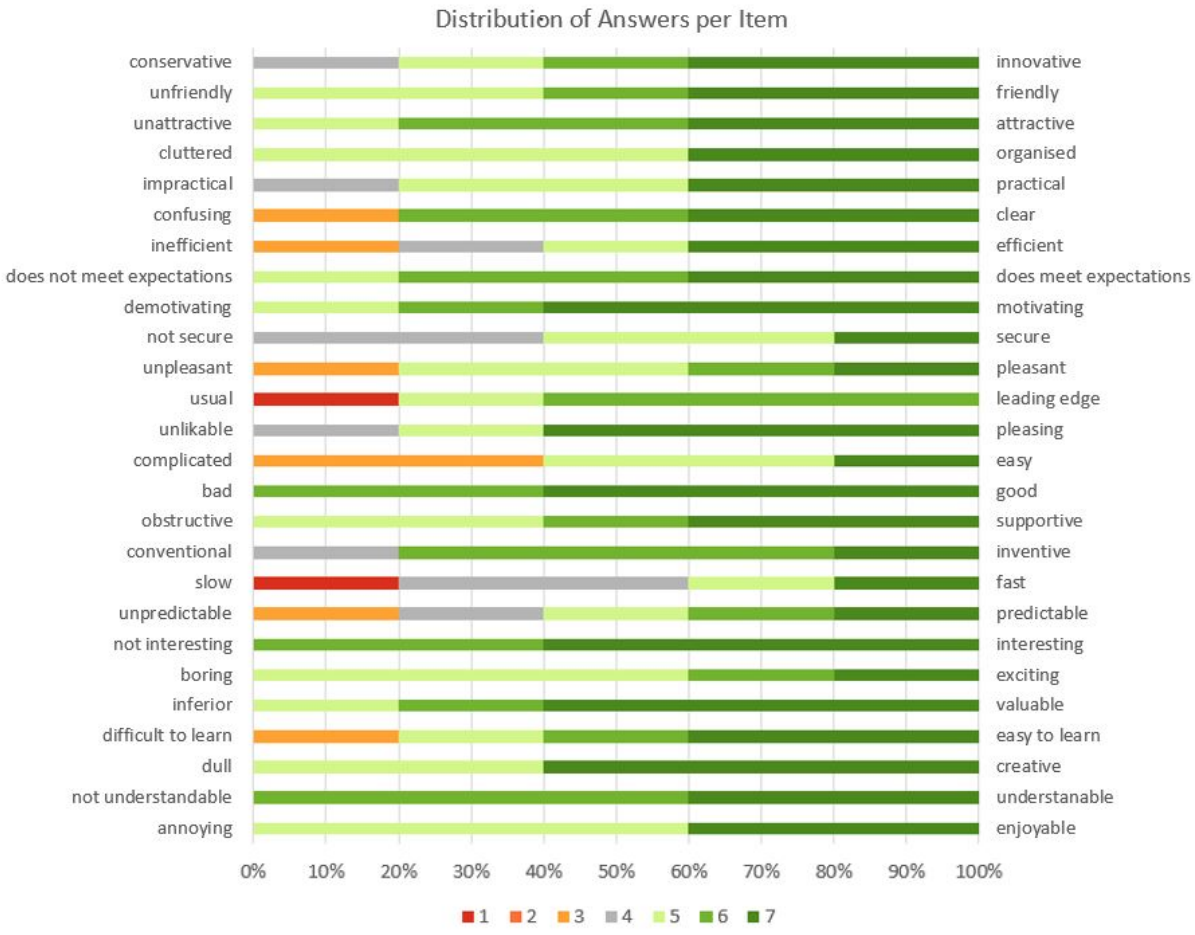


Figure 5.5: Diagram visualising the answer distribution. The values 1 to 7 correlate directly with the rating values -3 to +3.

Benchmark

The UEQ data analysis tool contains data of 21.175 products from 468 studies concerning different products. These are used as reference values to compare the results of the evaluated product to. As these reference products come from different backgrounds, such as business software, web pages, and social networks, this comparison only allows conclusions about the relative quality of the prototype.

Comparing the mean of the different scales, table 5.1 shows how the mean of this evaluation is always placed within the top 50%, most often even the top 10% of all evaluations.

Scale	Mean	Comparison	Interpretation
Attractiveness	1,97	**Excellent**	In the range of the 10% best results
Perspicuity	1,60	**Above Average**	25% of results better, 50% worse
Efficiency	1,20	**Above Average**	25% of results better, 50% worse
Dependability	1,55	**Good**	10% of results better, 75% worse
Stimulation	2,25	**Excellent**	In the range of the 10% best results
Novelty	1,65	**Excellent**	In the range of the 10% best results

Table 5.1: Comparison of the prototype to other products in the benchmark.

Figure 5.6 puts the result of the prototype in relation to the 21.175 other products provided by the benchmark analysis. The coloured areas indicate the values of the other products. It shows that the average of the products is always above or equal to 0.5, for example. The calculated mean values for this prototype are always placed above average. The mean of *effectiveness* being the lowest rating scale for the evaluated prototype and the *stimulation* scale being the highest rated scale.

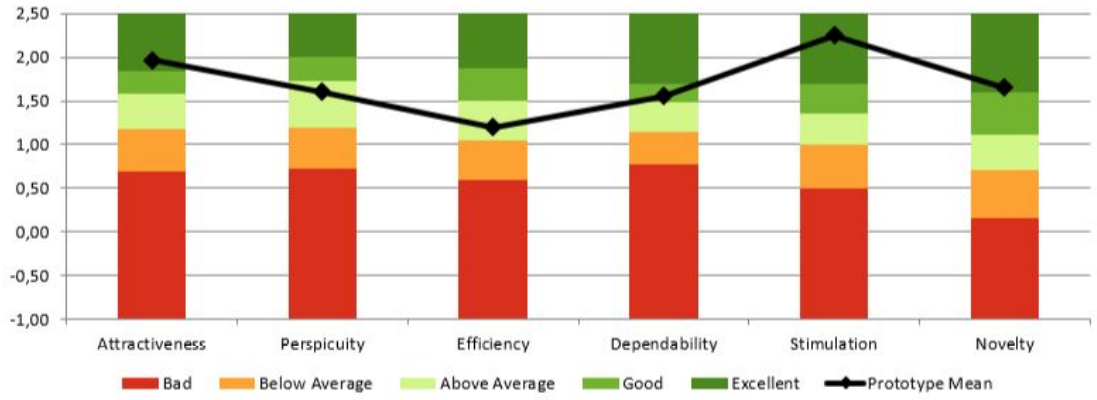


Figure 5.6: Diagram showing how the mean grading of the prototype always lies in the top half of all recorded product ratings.

5.1.5 Individual Feedback

This subsection summarises the semi-structured interviews conducted with each test subject after the usage of the prototype. The duration of the conversations ranged from almost 17 minutes to about 25 minutes each. The questionnaire, which collected demographic and experience data, showed that all participants had previous experience in using touch-sensitive tablets. However, only 3 out of 5 participants knew their way around 3D models, and only 2 people had basic prior experience with *Mixed Reality* (MR) applications and devices.

Participant 1

Participant 1 (P1) is currently a Ph.D. student who's research focuses on image processing. P1's vision was corrected to normal. P1 had worked with 3D models for several

years, but had little to no experience with MR applications. Although P1 thought the tablet was a good control, it felt heavy over time. Unlike the other participants, P1 chose to use the prototype while seated, positioning the model fairly high at chest level as can be seen in figure 5.7.

Figure 5.7: P1 seen in a seated position. The monitor in the background shows the exploration mode is currently active.

P1 stated both, the function to freeze a section plane and the function to take snapshots, were important to get insights into the inner structure of the 3D model. The tablet allowed easy spatial mapping, intersection with the 3D model resulting in frozen section planes or snapshots, and alignment of such created snapshots. P1 mainly worked with snapshots, but they were perceived as to be positioned too far away. Therefore, the alignment function was often used to get a closer look. Up close, the resolution of the internal data image and *Head Mounted Display* (HMD) was poor and P1 stated the combination made it difficult to see details. Of all the functions, slicing the model was most preferred as it helped to capture the internal information best in combination with

the 3D object. The neighbourhood function, on the other hand, was hardly used because P1 perceived the change between the neighbouring slices as not large enough to be easily visible. Furthermore, due to the small step size, the direction in which the inspection plane moved was not understandable. As P1 initially had difficulties understanding the prototype and memorising the different functions, it was suggested to integrate a tutorial into the prototype and give him the possibility to play videos. It was also proposed to give the user the possibility to adjust the step size of the neighbourhood inspection to make it more comfortable.

Participant 2

Participant 2 (P2) is currently a master student who's research focuses on information technology security on hardware devices. P2's vision was corrected to normal. P2 had a lot of experience with virtual environments and several years of practice in 3D modelling, and stated that all the controls were understandable and intuitive. Compared to the mechanical controllers which are commonly used to interact in an MRE, the touch controller felt innovative with its haptics and offered more interaction variety. For example, the creation of snapshots, which felt unique in its functionality. This and the alignment function were often used to keep track of the layout of the object. The slice neighbour inspection feature was also useful, although a step size calibration option would have been beneficial. When comparing the cutting and snapshot functions, cutting was found to be more helpful in understanding the internal structure, while snapshot provided additional information with respect to the unaltered external surface. While the rotation function was hardly used because it sometimes seemed confusing, mapping, snapshots, and alignment were the most popular features. P2 suggested highlighting the outline of the model in the snapshot to provide better contrast and also providing some kind of measurement information to inform about the dimension and scale of the model and the snapshots.

Participant 3

Participant 3 (P3) is working as a research assistant, researching image processing of ultra sound images. P3 had a normal eye vision and had no need for eye correction. The participant had three years of experience modelling three-dimensional data, but was not familiar with AR. P3 was the only participant who said that the tablet felt unnatural and the prototype required "too much interaction". While the weight of the HHD was fine, an "easier" interaction, such as the usage of hand recognition, was expected rather than the chosen tangible approach. While the rotation and resizing functions were predictable, the mapping would have been expected to move the object upwards when the tablet was rotated upwards, rather than synchronising the rotation. P3 explained, while freezing snapshots helps to understand the internal structure and is a preferred method to view the object and its interior from different angles, snapshots require much more interaction and are therefore less convenient. Apart from the instructional part, snapshots were therefore not used in the free exploration mode. In general, the cutting function was seen as beneficial, as conventional 3D programmes require additional actions to be explored, in contrast to the prototype. Nevertheless, the sense of the prototype, based on the use of a touch-sensitive and spatially capable tablet as a controller, was felt to involve too

much movement and general interaction. The use of eye contact and hand recognition was expected. It would have felt more natural to grab the 3D object with the hand to reposition it or cut it with a "knife"-like gesture. Neighbourhood inspection would also have been easier if the user could grab the origin plane of the selected snapshot and move it within the object instead of having to tilt the HHD.

Participant 4

Participant 4 (P4) is a bachelor student who studies information systems and has normal eye vision with no need for correction. P4 had no previous experience with *Augmented Reality* (AR) or 3D models. However, P4 quickly became familiar with the handling of the prototype and later stated that it was very well structured and comfortable to use. Nevertheless, the weight of the tablet was felt to be acceptable for short-term usage, but it could become heavy over time. In addition, it was not always easy to hold, e.g. when performing the turning movement, the hand position had to be changed to keep the HHD steady. Freezing the cutting plane was preferred to taking snapshots because in 3D the changes can be seen from all sides and the object looks "almost real". P4 said, "It felt good to cut something in reality. It was exactly how I would do it in real life. It didn't feel like it was not in reality." Although freezing the cutting planes was preferred, the snapshot function was also thoroughly explored. The placement of snapshots in relation to the direction of the swipe was found to be intuitive, although P4 sometimes confused swiping inwards with swiping outwards when working with snapshots or returning from a mode. When aligning all snapshots around the tablet, the HHD had to be held far away from the body to see all images as can be seen on figure 5.8. One thing which was special for P4 was that the position of the snapshots could still be changed after the initial placement and the selected snapshot was displayed on the HHD. Overall, P4 enjoyed the prototype and was impressed with the cutting and snapshot actions, which gave the impression of real interactions. It was suggested to cut the model at the position of the origin of a snapshot when it was selected. When deselected, the model could return to its original shape. This would help to see a better connection between the 2D image and the 3D model.

Figure 5.8: P4 holding the HHD with one arm stretched out to inspect aligned snapshots around the tablet.

Participant 5

Participant 5 (P5) is currently a Ph.D. student with a focus on the usage of handheld tablets which utilise stereoscopic methods. P5's vision was corrected to normal. P5 had no previous experience modelling three-dimensional data, but had already used *Virtual Reality* (VR) applications for several hours. P5 felt that using a tablet as a controller is very useful and a lot of time could be spent with it in a comfortable position without it being too heavy. One point which has always bothered P5 about VR has been that everything looks real but didn't feel real. Conventional controllers, for example, are alienating because they don't feel organic. The tablet, on the other hand, felt like a shape which P5 was used to, and it helped make the cutting feel more realistic, like the user could actually interact with an object. Cutting the 3D model felt very intuitive, "almost like using a knife". Comparing freezing the cutting plane with creating snapshots, the latter felt more like guessing and checking, which is why the former was preferable. The creation of snapshots was perceived as strange, as it required the user to take their eyes off the model. Nevertheless, the connection between 2D and 3D was easy to understand due to the positioned origin plane for each respective snapshot. The neighbourhood inspection feature was used a lot and although it was good to step through the 3D object, it felt very "snappy". In general, P5 felt using the prototype was "very interesting" as similar technologies were known from movies or similar research and it was "cool to interact with it". It was asked whether the bimodal menu could be merged, as the separation into exploration and selection did not seem necessary.

5.2 Discussion

The conducted user study was preliminary, using only five participants for the evaluation, meaning to collect first impressions and finding aspects about the concept and design of the prototype to improve. Overall, the feedback about the prototype and its different functionalities was very good, and many different opinions and suggestions could be collected. This section discusses all these findings, first summing up the information collected by the UEQ before going into detail of the individual opinions of all test subjects collected during the interview part, ending with the discussion about possible improvements.

The results shown by the UEQ were excellent and always above average when comparing it to other products, though it has to be kept in mind the used benchmark compares the prototype to different types of other applications only allowing a conclusion about its relative quality. From all items, it was described the most as good, interesting, motivating and understandable.

Besides all the positive findings, the UEQ results showed the prototype could be faster. This might be due to the used hardware which can easily be exchanged. In addition, some participants perceived the prototype as complicated. Comparing this point with the statements made in the interviews, it is clear this refers mainly to the beginning of the use of the prototype. The operations became clear after they had the chance to explore the prototype themselves.

The answer distribution of the UEQ showed, a divergence of opinions on the items "leading edge" and the fastness of the prototype. This can be explained by the previous experience with MR applications and 3D modelling the respective participants had. While people who had experience working with such objects might perceive the prototype as slow, amateur users might not feel the same way. Same can be explained with leading edge, though only one person did perceive the prototype as usual while all other leaned into the leading edge direction.

Going more into detail about the different opinions collected during the interviews helped explaining some of the tendencies displayed by the UEQ results.

The usage of a handheld tablet as means of interaction was generally well accepted. 4 out of 5 participants liked the HHD as an input device for an AR application. They all said it is good, though 2 of the 4 felt it gets heavy after some time. The one person (P3) not agreeing, stated that the HHD did not feel natural, and would have preferred less interaction and the use of hand recognition allowing the grabbing of virtual objects. This feeling could be due to the fact that this test person had the least experience (3 months) with the handling of a tablet and was not yet familiar enough with it as an input device. Apart from the criticism, two participants (P2, P5) who already worked with virtual environments stated, they preferred the tablet for such interactions over the conventional mechanical controllers, as it has a familiar shape, allows tangible interaction, and feels more realistic and natural.

From all interaction features, three test subjects (P1, P3, P4) stated they preferred the freezing of the cutting plane the most. It made the interaction feel "real" and help understand the object more than any other feature. Nevertheless, it was said the snapshots as well as the cutting of the object are important to understand and explore the

3D model. The context of queried two-dimensional data (snapshots) in form of an origin plane, was clear to all participants.

5.2.1 Suggestions

Apart from the positive feedback calling the prototype "understandable" (P2), "realistic" (P4, P5), and "intuitive" (P2, P4, P5), there were many points for improvement. One which was mentioned by four test subjects (P1, P2, P3, P5) was the movement of the origin plane of the snapshot when inspecting neighbour slices. This plane is moved using tilt gestures, though it was noted, the jump between the neighbours is not big enough. Therefore, it should be adjustable, allowing the user to configure the step size.

It was also mentioned that the resolution of the intersection images is lacking and the prototype is slow, which has already been picked up by UEQ. These issues could be addressed by using different hardware. A PC with a stronger graphic card and an HMD with better resolution might be able to solve this point of criticism. Concerning hardware, as already mentioned two participants felt like the tablet might get heavy with time. As seen in subsection 4.1.1, the prototype uses a combination of tracking device strapped to the HHD to determine its location in the tracking space. To get rid of some weight, this tracking device could be removed and the position determined using image recognition, making the HHD lighter.

Another issue mentioned by two participants (P1, P2), was the snapshot colour, which is too dark and the object surface, which should be highlighted. This was only perceived by these two test subjects, as the snapshots are programmed to always look at the user (billboarding). This way the user can move around the environment while the snapshots always face them. Unfortunately, the position of the light source within the AR scene can lead to shadows being applied to the planes depending on their angle. This is a good point to improve the visualisation of snapshots.

One participant stood out in comparison to the others when mentioning, that the prototype requires too much interaction using the tablet and wished for hand gestures. This could be a possible addition to the prototype. In subsection 2.4.1, a brief overview of mid-air gestures was given. Some of its drawbacks, such as the lack of tactile feedback, could be avoided when using it in combination with a touch-sensitive tablet. In this case, the use cases of the hand interactions need to be weight carefully. When using hand gestures, the tablet is only held in one hand, which might cause additional problems due to its weight. Therefore, the combination of gestures in the air and touch gestures on an HHD might be best suited when using a smaller HHD. This should allow some of the benefits of touch gestures, such as tactile feedback, while allowing the user to use the second hand for mid-air movements without the HHD being too heavy.

Chapter 6

Conclusion

The focus of this master thesis lies on the interaction with volumetric data and how to design it in such a way that is both natural and easy for the user to handle. The research conducted has shown that three-dimensional objects can be explored well by allowing the user to open the model and take a look at its internal structure. This can be done by removing parts of the object and thus exposing its insides. Taking snapshots is also a good way to store information about certain parts for later viewing.

While there are many different input devices, only a few of them allow the user to perform such actions in a natural way. Usually, conventional mechanical controllers are used for any interaction in *Augmented Reality* (AR). These can feel alienating and unnatural and do not offer intuitive control for the mentioned explorations. Mid-air gestures, on the other hand, are not designed to be used over a long period of time and lead to fatigue (gorilla-arm-effect). Compared to other devices, the use of a handheld, touch-sensitive tablet showed the most advantages in terms of the use cases. In addition to tactile feedback, it is capable of spatial and touch based input as well as output. It is an off-the-shelf device that is easy to hold and use and is already integrated into many people's lives and has therefore become a familiar form. Furthermore, its shape allows it to be used in a tangible way that mimics a cutting plane when interacting with the three-dimensional object.

The created prototype, which uses such a *handheld device* (HHD), was tested as part of a small user study. As discussed, the prototype implementation received mixed feedback, most of which was positive. Revisiting the posed research questions, has shown that the design and implementation of the proposed concept makes the retrieval and comparison of information of a data set understandable. The usage of a spatially aware, touch-sensitive HHD with an attached cutting plane to explore three-dimensional data was perceived as realistic and intuitive. It was well accepted by the bigger part of the participants, although some changes can be made with regards to its total weight. The connection between the three-dimensional data and the two-dimensional exploration data was experienced to be comprehensible by all participants. However, the information obtained in this study is not sufficient to draw statistical conclusions. They merely show that the concept is going in the right direction and give suggestions as to where improvements are still needed.

6.1 Outlook

Conducting the user study has helped to identify some points in the concept of this prototype that could be improved. However, since the study was carried out with only five test subjects, it would be particularly interesting to conduct another study with more participants and better hardware. Apart from this, there are several points that could complement the prototype well. As mentioned by several users, it would be beneficial if the user could manually configure the step size between slices when iterating through the neighbourhood information of a snapshot. Image capture and processing could be added to track the position and rotation of the tablet. This would allow the tracking sensor to be removed, reducing the weight of the HHD and making it more manageable. Moreover, it would enable the addition of mid-air gestures, as suggested by one participant. The point of collaboration, which was defined as a non-target in this master thesis, could also become interesting in the future. In addition to multiple users working on the same model, inspecting a number of models simultaneously could also be beneficial. Two or more models could be positioned next to each other and cut in the same places to compare them.

Appendix A

List of Acronyms

AR Augmented Reality
BMP Bitmap
CPU Central Processing Unity
CT Computer Tomography
DOF Degrees of Freedom
FBX Filmbox
FOV Field of View
FPS Frames per Second
GPU Graphics Processing Unit
GUI Graphical User Interface
HHD Handheld Device
HMD Head Mounted Display
HUD Head Up Display
MR Mixed Reality
MRE Mixed Reality Environment
MRI Magnetic Resonance Imaging
PC Personal Computer
PNG Portable Network Graphics
RAM Random Access Memory
ROI Region of Interest
RSP C# Compiler Response File
SDK Software Development Kit
STL Standard Triangle Language
VR Virtual Reality

Appendix B

Evaluation Documents

Information according to Article 13 GDPR

The protection of personal data is of particular concern to us. We therefore process personal data exclusively in accordance with the provisions of the General Data Protection Regulation (GDPR) and applicable national legislation. In this overview, we provide information about the most important aspects of data processing.

Purpose of data processing
The aim of the preliminary study is to test various interaction possibilities with three-dimensional data using a touch-sensitive tablet in **augmented reality**. The following data will be collected from data subjects and subsequently processed:

- Demographic data
- Answers to standardised questionnaires
- Answers to oral questions
- Video and audio recordings of the study conducted
- Screen recordings of the augmented scene of the study conducted

Rights of data subjects
In the area of scientific or historical research purposes or statistical purposes, exceptions to the rights of data subjects may be provided for under Article 89(2) of the GDPR and Section 2d(6) of the Research Organisation Act (FOG), insofar as the achievement of research purposes is likely to be rendered impossible or seriously impaired.

We inform you of the right to lodge a complaint with the Austrian Data Protection Authority, Barichgasse 40-42, 1030 Vienna, telephone: +43 1 521 52 - 2569, e-mail: dsb@dsb.gv.at as the local supervisory authority or with the otherwise competent supervisory authority (e.g. of the place of residence or place of work) in accordance with Article 77 GDPR.

Contact details for questions regarding data protection:
Janine Mayer
Softwarepark 11, 4232 Hagenberg, Austria
Janine.Mayer@fh-hagenberg.at

☐ I agree that a non-anonymised image of me from the recorded video material may be published in scientific publications

I confirm with my signature that I have read and understood the information above.

_____ _____

Place, Date Signature

_____

Name

Preliminary Study
Volumetric Data Interaction

The purpose of this preliminary study is to evaluate the functional prototype which has been created in the course of a master thesis. This prototype uses Augmented Reality (AR) to visualize a three-dimensional data set in the user's environment. The use of a spatially aware, touch-sensitive handheld tablet allows for manipulation and exploration of said data set. The prototype is developed in a way which allows minimal eye contact with the input device to avoid arm fatigue.

Purpose

- Identifying interactions and manipulations which are intuitive or hard to use.
- Finding which features are easy to understand and which are not.
- Finding weaknesses and strengths in the design of this prototype.

Structure

This user evaluation is separated into multiple parts and will take up to **one hour**.

Part 1	Introduction and demographic questionnaire	5 minutes
Part 2	Prototype usage	
	a. Introduction & Tutorial	8 minutes
	b. Exploration using instructions	12 minutes
	c. Free exploration	10 minutes
Part 3	Follow up questions	
	a. Questionnaire about the usage	3 minutes
	b. Semi-structured Interview	20 minutes

Information about study risks

The usage of a Head Mounted Device to show Augmented Reality may lead to discomfort. If you feel unwell, dizzy, or simply do not want to continue with the study please tell the person in charge. The participant can leave this study at any time without consequences.

Part 1 - Questionnaire

Name				
Age				
Gender	O female	O male	O other	O do not disclose
Need of eye correction	O yes	O no		
Use of eye correction	O yes	O no		
Colour vision deficiency				

Have you had any previous experience using an AR application? O Yes O No

If yes, how long, please describe: _____

_____

_____

Have you had any previous experience using a touch-sensitive tablet? O Yes O No

If yes, how long, please describe: _____

_____

_____

Have you had any previous experience exploring 3D data? O Yes O No

If yes, how long, please describe: _____

_____

_____

Part 2 - Prototype Usage

The prototype was designed in a way to allow the user to interact with and explore three-dimensional data in augmented reality. The spatial and touch-sensitive capabilities of the handheld tablet allow the user to perform such operations.

The study course is designed in a way to allow the user to explore the different functionalities step by step. After going through these functionalities together, the participant is free to explore the model using the prototype further if wanted.

Part A – Introduction

First the general prototype is introduced, followed by short videos which display the usage of the different available features.

Tutorial available here:

https://docs.google.com/presentation/d/19bP3DuU_p4yXLniWSFz4XfeaMjKH1Ve6/edit?usp=sharing&ouid=116127959025007089702&rtpof=true&sd=true

Part B – Instructions

1. Rearrange the model in your environment by adjusting its position, rotation, and size.
2. Create snapshots and position them in the room.
    a. Align the snapshots around the tablet and compare them.
    b. Select a snapshot and investigate its neighbours (layer by layer).
3. Cut the model at specific positions.
4. Remove all objects and reset the model.

Part C – Free Usage

Please manipulate and explore the dataset now in a way you want. Feel free to use any feature you learned before. You can take as long as you want. Try to understand the model.

Part 3 – Questionnaire & Interview

Please assess the prototype by filling out the following questionnaire. This questionnaire consists of pairs of contrasting attributes that may apply to the product. The circles between these attributes represent graduations between the opposites. Please express your agreement with the attributes by ticking one circle per line that most closely reflects your impression.

	1	2	3	4	5	6	7		
annoying	○	○	○	○	○	○	○	enjoyable	1
not understandable	○	○	○	○	○	○	○	understandable	2
creative	○	○	○	○	○	○	○	dull	3
easy to learn	○	○	○	○	○	○	○	difficult to learn	4
valuable	○	○	○	○	○	○	○	inferior	5
boring	○	○	○	○	○	○	○	exciting	6
not interesting	○	○	○	○	○	○	○	interesting	7
unpredictable	○	○	○	○	○	○	○	predictable	8
fast	○	○	○	○	○	○	○	slow	9
inventive	○	○	○	○	○	○	○	conventional	10
obstructive	○	○	○	○	○	○	○	supportive	11
good	○	○	○	○	○	○	○	bad	12
complicated	○	○	○	○	○	○	○	easy	13
unlikable	○	○	○	○	○	○	○	pleasing	14
usual	○	○	○	○	○	○	○	leading edge	15
unpleasant	○	○	○	○	○	○	○	pleasant	16
secure	○	○	○	○	○	○	○	not secure	17
motivating	○	○	○	○	○	○	○	demotivating	18
meets expectations	○	○	○	○	○	○	○	does not meet expectations	19
inefficient	○	○	○	○	○	○	○	efficient	20
clear	○	○	○	○	○	○	○	confusing	21
impractical	○	○	○	○	○	○	○	practical	22
organized	○	○	○	○	○	○	○	cluttered	23
attractive	○	○	○	○	○	○	○	unattractive	24
friendly	○	○	○	○	○	○	○	unfriendly	25
conservative	○	○	○	○	○	○	○	innovative	26

Semi-structured Interview

This semi-structured interview will be recorded in order to best understand and evaluate the answers of the participant.

Introduction

How did you like the usage of the prototype? (Intuitive, bothersome, complicated …)

How did you perceive the tablet as a mean of interaction with the two- and three- dimensional data? How did you like using touch- and spatial input to interact with the model?

Have you felt any discomfort while using the prototype? Tablet (fatigue, uncomfortable)?

Have you had any trouble executing an action? Did any interaction do something different than expected? E.g., did you expect a swipe to do something different than a double tap

Object Manipulation and Removal

How was the handling of the object in general? Has everything worked as expected?

Slicing the 3D Object

How was using the tablet as a cutting plane? (Intuitive, weird, easy)? Did you always know where the cutting plane was?

Did cutting away parts of the prototype help understand the internal structure?

Snapshots

How was creating snapshots? Was the positioning when creating helpful, confusing, hard?

Did the snapshots help understand the internal structure of the 3D model?

Did aligning and misaligning help comparing the snapshots, or did you prefer the snapshots in your environment?

Did the visualisation of the snapshot origin within the 3D model help understand the connection between model and snapshot?

Did you always know which part of the model was concerned when investigating a snapshot?

How did you like the inspection of snapshot neighbours? (Easy, intuitive, hard, confusing) Did it help to understand the internal structure?

Conclusion

What did you like most? (Why?)

What did bother you most?

What do you think is lacking to enhance these interactions?

List of Figures

2.1 Positioning a clipping plane with a tracked panel [71]. 4
2.2 Positioning a clipping plane with a tracked pen [86]. 4
2.3 Creating snapshots from cutting planes displaying CT data [15]. 5
2.4 Slicing a 3D data set using passive projection on a Plexiglas® [22]. . . . 5
2.5 Touch input manipulating stereoscopically visualised data [52]. 6
2.6 Sereno et al. collaboratively manipulate data using touch input [73]. . . 6
2.7 TabletInVR allows the user to interact with a virtual object using a touch-sensitive tablet [78]. a) Rotations are executed by performing a twist gesture on the display. b) The selected object can be clipped by handling the tablet as a "knife". 7
2.8 Exploration of a three-dimensional data set in AR [55]. 8
2.9 Cubic Mouse with tree perpendicular rods which can be used for the translation of cutting planes [33]. 9
2.10 The GlobeFish is a 6-DOF controller with a 3D trackball [31]. 10
2.11 GlobePointer which is based on the GlobeFish [32]. 10
2.12 The GlobeMouse is a tracked 3D ball incorporated on a 3D translation controller [31]. 11
2.13 The SquareBone is a two-handed device, which consists of two 6-DOF SpaceMouse devices [39]. 11
2.14 The roller mouse can be used in three dimensions [83]. 11
2.15 Eye of Ra can switch between 2D and 3D interaction [14]. 11
2.16 Personal Interaction Panel (PIP) introduced by Szalavari et al. in 1997 [80]
a) The PIP consists of a tracked panel and pen which are augmented using an HMD. b) The PIP can be used to hold interaction tools, c) or to slice virtual objects and display internal structures. 13
2.17 VESAD maps virtual windows to the user's mobile device [62]. 15
2.18 The formative interview conducted by Ren et al. concluded following examples [69] a) The smartphone can be shaken to open a window. b) An application can be opened in AR when pressing the app icon while executing a shake gesture. c) If the user holds the edge of the smartphone when it collides with the AR window, the window is to be closed. 16
2.19 STREAM allows the user to use a tablet for a fluid combination of 2D and 3D data [38]. 20
2.20 Ray casting using a pointing gesture [61]. 22

2.21 The Framing Hands Technique [65]. 22
2.22 Flexible Pointer [63]. 23
2.23 iSith ray cast selection [88]. 23
2.24 3D wireframe lens and magnifier [8]. 24
2.25 X Ray volumetric lens using a defined frustum [84]. 24

3.1 Setup as depicted by Mayer et al. [59]. 26
3.2 Interaction of the components, with arrows indicating dependency. . . . 27
3.3 CT Scan of a concert guitar done by *Research Group Computed Tomography*. 28
3.4 Diagram describing the different interaction states. 31
3.5 Rotations can be executed using a twist gesture as depicted by Mayer et al. [59]. 33
3.6 The resizing of an object can be executed using a pinch gesture as depicted by Mayer et al. [59]. 33
3.7 The calculated volumetric cut is displayed on the tablet while the virtual boundary of the virtual object (blue, visualised using the HMD) is still visible. 35
3.8 Clipped virtual surface model (blue) with frozen cutting planes displaying the volumetric data superimposed on the clipped part. 36
3.9 Snapshots are a momentary recording of a cutting slice and can be placed in the user's environment. The swipe direction tells the system in which direction the snapshot should be placed. 37
3.10 All snapshots are placed at specified locations in the user's environment. Occlusion is possible. The grab gesture triggers the alignment of existing snapshots around the HHD. 38
3.11 Bi-sectional menu parted into selection and exploration mode. 39

4.1 Interconnection between hardware devices and user. 44
4.2 The clip holder is mounted on the back of the tablet, only touching the top and bottom. 45
4.3 The connection part (white) connects tracking marker and tablet clip holder. 45
4.4 Data processing for its use for the prototype. 47
4.5 Image of a slice along the x-axis of the volumetric data. 48
4.6 Image of a slice along the y-axis of the volumetric data. 48
4.7 Image of a slice along the z-axis of the volumetric data. 48
4.8 Object with the iso-value chosen too low (12,533) showing disturbances on the surface. 49
4.9 Object with the iso-value chosen too high (13,100) showing many holes on the surface. 49
4.10 Selection of number of iso-surfaces using ImageVis3D. 50
4.11 The images on the left side show the imported model from ImageVis3D. The images on the right show the final data set after being simplified and manually adjusted using blender. 51
4.12 NetworkMessage and all derived message classes. 52

4.13 The process of initialisation within the host class, as well as the sending and receiving of network messages. 54
4.14 Interconnection between main components and main classes. 55
4.15 UI screens rendered in the different states of the prototype. 58
4.16 The main mode overlay depicts how a left tap starts the selection mode and a right tap starts the exploration mode. In addition, the reset function, which is executed using shakes, and the return functions are depicted. 59
4.17 The left image shows the UI for the selection mode. During this mode the user can either double tap when the selection ray is intersecting an object or return to the main mode. The image on the right depicts some of the possible actions when the selected mode is active. Not all possible actions are presented to allow a better overview. 60
4.18 Interactions when the user has entered the selection mode. 61
4.19 The UI for the exploration mode only depicts the two main functionalities: the cutting of the snapshot and the creating of a snapshot. The alignment of created snapshots has been left out for a better overview to focus on the main aspects of exploration. 62
4.20 Interactions when the user has entered the exploration mode. 63
4.21 The temporary cutting plane is positioned slightly before the tablet overlay. This gives the user a visual hint of the surface's current position and providing better intuition of the tablet's position within the model. . . . 64
4.22 Class diagram of the Slicer and the SliceListener classes. 65
4.23 Example of a permanently changed model using the permanent cutting functionality of the prototype. The intersected part has been removed and a material containing the calculated volumetric data texture has been added to the intersecting face. 66
4.24 Model class with all properties and methods. 66
4.25 The section plane is shown in blue, while the points used to determine the section boundary with the 3D object are shown in red. The left image depicts how the points start at the edges of the plane and move towards its centre. In the right image the points have stopped on colliding with the boundary of the slicable object. 67
4.26 All snapshots are positioned in the user's environment. 70
4.27 All snapshots are aligned around the HHD. 70
4.28 The neighbour slice is shown on the tablet overlay instead of the selected snapshot which allows a direct comparison to the original snapshot. . . 72
4.29 The blue intersections described on the left side show valid positions for the cutting plane. The plane intersects the object completely. The red intersections seen on the right are invalid because the positions never completely cut an entire axis. 73

5.1 Example of a pair of aspects and the seven-level gradation which the test subjects can choose from. 77
5.2 Real world environment in which the study was conducted. 78
5.3 Diagram cumulating the mean of all ratings grouped the main scales. . . 79
5.4 Mean, median, variance, and standard deviation of the ratings by scale. 79

5.5 Diagram visualising the answer distribution. The values 1 to 7 correlate directly with the rating values -3 to +3. 81
5.6 Diagram showing how the mean grading of the prototype always lies in the top half of all recorded product ratings. 82
5.7 P1 seen in a seated position. The monitor in the background shows the exploration mode is currently active. 83
5.8 P4 holding the HHD with one arm stretched out to inspect aligned snapshots around the tablet. 86

List of Tables

2.1 Generic input devices overview. 12
2.2 Panels and tablets overview. 14

3.1 Available actions in the main menu. 40
3.2 Available actions in selection mode. 41
3.3 Available actions in exploration mode. 42

4.1 Resolution of the volumetric dataset. 47

5.1 Comparison of the prototype to other products in the benchmark. . . . 82

References

Literature

[1] Denis Amselem. “A Window on Shared Virtual Environments”. *Presence: Teleoperators and Virtual Environments* 4.2 (Jan. 1995), pp. 130–145 (cit. on pp. 13, 14).

[2] Ian G. Angus and Henry A. Sowizral. “Embedding the 2D Interaction Metaphor in a Real 3D Virtual Environment”. In: IS&T/SPIE’s Symposium on Electronic Imaging: Science & Technology. Ed. by Scott S. Fisher, John O. Merritt, and Mark T. Bolas. San Jose, CA, Mar. 30, 1995, pp. 282–293 (cit. on pp. 12, 14).

[3] Christoph Anthes, Rubén Jesús Garcıa-Hernández, Markus Wiedemann, and Dieter Kranzlmüller. “State of the art of virtual reality technology”. In: *2016 IEEE aerospace conference*. IEEE. 2016, pp. 1–19 (cit. on pp. viii, ix).

[4] Ronald T Azuma. “A survey of augmented reality”. *Presence: teleoperators & virtual environments* 6.4 (1997), pp. 355–385 (cit. on pp. viii, ix, 1).

[5] Lonni Besançon, Mickael Sereno, Lingyun Yu, Mehdi Ammi, and Tobias Isenberg. “Hybrid Touch/Tangible Spatial 3D Data Selection”. *Computer Graphics Forum* 38.3 (June 2019), pp. 553–567 (cit. on pp. 21, 33).

[6] Verena Biener, Daniel Schneider, Travis Gesslein, Alexander Otte, Bastian Kuth, Per Ola Kristensson, Eyal Ofek, Michel Pahud, and Jens Grubert. “Breaking the Screen: Interaction Across Touchscreen Boundaries in Virtual Reality for Mobile Knowledge Workers”. *IEEE Transactions on Visualization and Computer Graphics* 26.12 (Dec. 2020), pp. 3490–3502 (cit. on p. 15).

[7] Eric A Bier. “Snap-dragging in three dimensions”. *ACM SIGGRAPH Computer Graphics* 24.2 (1990), pp. 193–204 (cit. on p. 10).

[8] Eric A Bier, Maureen C Stone, Ken Pier, William Buxton, and Tony D DeRose. “Toolglass and magic lenses: the see-through interface”. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 1993, pp. 73–80 (cit. on pp. 23, 24).

[9] Eric A. Bier and Maureen C. Stone. “Snap-Dragging”. In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’86. New York, NY, USA: Association for Computing Machinery, 1986, pp. 233–240 (cit. on p. 10).

[10] O. Bimber, L.M. Encamacao, and D. Schmalstieg. “Real mirrors reflecting virtual worlds”. In: *Proceedings IEEE Virtual Reality 2000 (Cat. No.00CB37048)*. 2000, pp. 21–28 (cit. on p. 4).

[11] Oliver Bimber, L Miguel Encarnaçáo, and Dieter Schmalstieg. “Augmented Reality with Back-Projection Systems using Transflective Surfaces”. In: *Computer Graphics Forum*. Vol. 19. 3. Wiley Online Library. 2000, pp. 161–168 (cit. on p. 4).

[12] Richard A Bolt. ““Put-that-there” Voice and gesture at the graphics interface”. In: *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*. 1980, pp. 262–270 (cit. on pp. 16, 21).

[13] Sebastian Boring, Marko Jurmu, and Andreas Butz. “Scroll, Tilt or Move It: Using Mobile Phones to Continuously Control Pointers on Large Public Displays”. In: *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group on Design: Open 24/7 - OZCHI ’09*. The 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group. Melbourne, Australia: ACM Press, 2009, p. 161 (cit. on pp. 17–19, 26, 38).

[14] A. Bornik, R. Beichel, E. Kruijff, B. Reitinger, and D. Schmalstieg. “A Hybrid User Interface for Manipulation of Volumetric Medical Data”. In: *3D User Interfaces (3DUI’06)*. 2006, pp. 29–36 (cit. on pp. 10–12).

[15] Alexander Bornik, Reinhard Beichel, Bernhard Reitinger, Georg Gotschuli, Erich Sorantin, Franz Leberl, and Milan Sonka. “Computer Aided Liver Surgery Planning: An Augmented Reality Approach”. *Proc SPIE* 5029 (May 1, 2003) (cit. on pp. 4, 5, 34, 35).

[16] Alexander Bornik, Bernhard Reitinger, Reinhard Beichel, Erich Sorantin, and Georg Werkgartner. “Augmented-Reality-Based Segmentation Refinement”. In: Medical Imaging 2004. Ed. by Robert L. Galloway Jr. San Diego, CA, May 5, 2004, p. 77 (cit. on p. 5).

[17] Douglas A Bowman. *Interaction techniques for common tasks in immersive virtual environments: design, evaluation, and application*. Georgia Institute of Technology, 1999 (cit. on pp. 16, 21, 34).

[18] John Brook. “SUS: a" quick and dirty" usability scale”. *Usability evaluation in industry* 189.3 (1996) (cit. on p. 15).

[19] Gerd Bruder, Frank Steinicke, and Wolfgang Sturzlinger. “To Touch or Not to Touch? Comparing 2D Touch and 3D Mid-Air Interaction on Stereoscopic Tabletop Surfaces”. In: *Proceedings of the 1st Symposium on Spatial User Interaction*. SUI ’13. New York, NY, USA: Association for Computing Machinery, July 20, 2013, pp. 9–16 (cit. on pp. 17, 18).

[20] Steve Bryson. “Virtual Reality in Scientific Visualization”. *Communications of the ACM* 39.5 (May 1996), pp. 62–71 (cit. on p. 8).

[21] Wolfgang Büschel, Patrick Reipschläger, Ricardo Langner, and Raimund Dachselt. "Investigating the Use of Spatial Interaction for 3D Data Visualization on Mobile Devices". In: *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. ISS '17: Interactive Surfaces and Spaces. Brighton United Kingdom: ACM, Oct. 17, 2017, pp. 62–71 (cit. on pp. 17, 18, 26).

[22] Alvaro Cassinelli and Masatoshi Ishikawa. "Volume slicing display." *SIGGRAPH ASIA Art Gallery & Emerging Technologies* 88 (2009) (cit. on pp. 5, 34).

[23] Marco Cavallo, Mishal Dolakia, Matous Havlena, Kenneth Ocheltree, and Mark Podlaseck. "Immersive Insights: A Hybrid Analytics System ForCollaborative Exploratory Data Analysis". In: *25th ACM Symposium on Virtual Reality Software and Technology*. VRST '19. Parramatta, NSW, Australia: Association for Computing Machinery, 2019 (cit. on p. 17).

[24] Li-Wei Chan, Hui-Shan Kao, Mike Y Chen, Ming-Sui Lee, Jane Hsu, and Yi-Ping Hung. "Touching the void: direct-touch interaction for intangible displays". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2010, pp. 2625–2634 (cit. on pp. 1, 17, 36, 39).

[25] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. "Tetrahedra based volume visualization". In: *Mathematical Visualization*. Springer, 1998, pp. 3–18 (cit. on p. 29).

[26] Aurélie Cohé and Martin Hachet. "Beyond the Mouse: Understanding User Gestures for Manipulating 3D Objects from Touchscreen Inputs". *Computers & Graphics* 36.8 (Dec. 2012), pp. 1119–1131 (cit. on pp. 18, 19, 32).

[27] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE". In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '93. Anaheim, CA: Association for Computing Machinery, 1993, pp. 135–142 (cit. on p. 9).

[28] Leonardo De Chiffre, Simone Carmignato, J-P Kruth, Robert Schmitt, and Albert Weckenmann. "Industrial applications of computed tomography". *CIRP annals* 63.2 (2014), pp. 655–677 (cit. on p. 28).

[29] George W Fitzmaurice, Shumin Zhai, and Mark H Chignell. "Virtual reality for palmtop computers". *ACM Transactions on Information Systems (TOIS)* 11.3 (1993), pp. 197–218 (cit. on pp. 13, 14).

[30] George W. Fitzmaurice. "Situated Information Spaces and Spatially Aware Palmtop Computers". *Communications of the ACM* 36.7 (July 1, 1993), pp. 39–49 (cit. on p. 13).

[31] Bernd Froehlich, Jan Hochstrate, Verena Skuk, and Anke Huckauf. "The globefish and the globemouse: two new six degree of freedom input devices for graphics applications". In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006, pp. 191–199 (cit. on pp. 9–12).

[32] B. Frohlich, J. Hochstrate, A. Kulik, and A. Huckauf. "On 3D Input Devices". *IEEE Computer Graphics and Applications* 26.2 (Mar. 2006), pp. 15–19 (cit. on pp. 9, 10, 12).

[33] Bernd Fröhlich and John Plate. "The Cubic Mouse: A New Device for Three-Dimensional Input". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '00*. The SIGCHI Conference. The Hague, The Netherlands: ACM Press, 2000, pp. 526–531 (cit. on pp. 9, 10, 12, 34, 35).

[34] Luigi Gallo, Giuseppe De Pietro, and Ivana Marra. "3D Interaction with Volumetric Medical Data: Experiencing the Wiimote". In: *Proceedings of the First International Conference on Ambient Media and Systems*. 1st International ICST Conference on Ambient Media and Systems. Quebec, Canada: ICST, 2008 (cit. on pp. 11, 12).

[35] Yves Guiard. "Asymmetric Division of Labor in Human Skilled Bimanual Action". *Journal of Motor Behavior* 19.4 (Dec. 1, 1987), pp. 486–517. pmid: 15136274 (cit. on p. 26).

[36] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. "Consumed endurance: a metric to quantify arm fatigue of mid-air interactions". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2014, pp. 1063–1072 (cit. on pp. 1, 17, 26, 34).

[37] Ken Hinckley, Randy Pausch, John C Goble, and Neal F Kassell. "A survey of design issues in spatial input". In: *Proceedings of the 7th annual ACM symposium on User interface software and technology*. 1994, pp. 213–222 (cit. on pp. 17, 21, 22, 36, 39).

[38] Sebastian Hubenschmid, Johannes Zagermann, Simon Butscher, and Harald Reiterer. "Stream: Exploring the combination of spatially-aware tablets with augmented reality head-mounted displays for immersive analytics". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–14 (cit. on pp. 19, 20, 26, 37, 38).

[39] Anke Huckauf, Alexander Speed, André Kunert, Jan Hochstrate, and Bernd Fröhlich. "Evaluation of 12-dof input devices for navigation and manipulation in virtual environments". In: *IFIP Conference on Human-Computer Interaction*. Springer. 2005, pp. 601–614 (cit. on pp. 9–12).

[40] Paul Issartel, Florimond Guéniat, and Mehdi Ammi. "A Portable Interface for Tangible Exploration of Volumetric Data". In: *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology - VRST '14*. The 20th ACM Symposium. Edinburgh, Scotland: ACM Press, 2014, pp. 209–210 (cit. on pp. 5, 26, 34).

[41] Robert JK Jacob. "Eye tracking in advanced interface design". *Virtual environments and advanced interface design* 258 (1995), p. 288 (cit. on pp. 15, 20).

[42] Eric Arthur Johnson. "Touch display–a novel input/output device for computers". *Electronics Letters* 1.8 (1965), pp. 219–220 (cit. on p. 17).

[43] A Kaufman. "Volume visualization: Principles and advances". *Course notes* 24 (1997) (cit. on pp. 1, 28).

[44] Regis Kopper, Felipe Bacim, and Doug A Bowman. "Rapid and accurate 3D selection by progressive refinement". In: *2011 IEEE symposium on 3D user interfaces (3DUI)*. IEEE. 2011, pp. 67–74 (cit. on pp. 23, 34).

[45] Panayiotis Koutsabasis and Panagiotis Vogiatzidakis. “Empirical research in mid-air interaction: A systematic review”. *International Journal of Human–Computer Interaction* 35.18 (2019), pp. 1747–1768 (cit. on p. 16).

[46] Wolfgang Krueger and Bernd Fröhlich. “Responsive workbench”. In: *Virtual Reality'94*. Springer, 1994, pp. 73–80 (cit. on p. 9).

[47] W. Kruger, C.-A. Bohn, B. Frohlich, H. Schuth, W. Strauss, and G. Wesche. “The Responsive Workbench: a virtual work environment”. *Computer* 28.7 (1995), pp. 42–48 (cit. on p. 9).

[48] Bireswar Laha and Doug A Bowman. “Identifying the benefits of immersion in virtual reality for volume data visualization”. In: *Immersive visualization revisited workshop of the IEEE VR conference*. Citeseer. 2012, pp. 1–2 (cit. on p. 8).

[49] Chi-Jung Lee and Hung-Kuo Chu. “Dual-MR: Interaction with Mixed Reality Using Smartphones”. In: *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*. VRST '18: 24th ACM Symposium on Virtual Reality Software and Technology. Tokyo Japan: ACM, Nov. 28, 2018, pp. 1–2 (cit. on pp. 17, 18).

[50] Jiandong Liang and Mark Green. “Geometric modeling using six degrees of freedom input devices”. In: *Proc. 3rd International Conference on CAD and Computer Graphics*. Citeseer. 1993, pp. 217–222 (cit. on pp. 21, 22).

[51] Ching-yao Lin, R Bowen Loftin, Ioannis A Kakadiaris, David T Chen, and Simon Su. “Interaction with medical volume data on a projection workbench”. In: *The Proceedings of 10th International Conference on Artificial Reality and Telexistence*. 2000, pp. 148–152 (cit. on pp. 1, 14, 29).

[52] David López, Lora Oehlberg, Candemir Doger, and Tobias Isenberg. “Tablet-based interaction for immersive 3d data exploration”. In: *Posters at the IEEE Conference on Visualization (IEEE VIS 2014, November 9–14, Paris, France)*. 2014 (cit. on pp. 6, 26).

[53] David López, Lora Oehlberg, Candemir Doger, and Tobias Isenberg. “Towards An Understanding of Mobile Touch Navigation in a Stereoscopic Viewing Environment for 3D Data Exploration”. *IEEE Transactions on Visualization and Computer Graphics* 22.5 (May 2016), pp. 1616–1629 (cit. on pp. 19, 26, 32).

[54] William E. Lorensen and Harvey E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. *ACM SIGGRAPH Computer Graphics* 21.4 (Aug. 1, 1987), pp. 163–169 (cit. on p. 29).

[55] Weizhou Luo, Eva Goebel, Patrick Reipschläger, Mats Ole Ellenberg, and Raimund Dachselt. “Exploring and Slicing Volumetric Medical Data in Augmented Reality Using a Spatially-Aware Mobile Device”. In: *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE. 2021, pp. 334–339 (cit. on pp. 7, 8, 34).

[56] Alessio Malizia and Andrea Bellucci. “The Artificiality of Natural User Interfaces”. *Communications of the ACM* 55.3 (Mar. 2012), pp. 36–38 (cit. on p. 34).

[57] Gary Marsden and Nicholas Tip. "Navigation control for mobile virtual environments". In: *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*. 2005, pp. 279–282 (cit. on pp. 17, 26).

[58] Asier Marzo, Benoît Bossavit, and Martin Hachet. "Combining Multi-Touch Input and Device Movement for 3D Manipulations in Mobile Augmented Reality Environments". In: *Proceedings of the 2nd ACM Symposium on Spatial User Interaction*. SUI '14. New York, NY, USA: Association for Computing Machinery, Oct. 4, 2014, pp. 13–16 (cit. on pp. 17, 18, 32).

[59] Janine Mayer, Stefan Auer, Judith Friedl, and Christoph Anthes. "Volumetric Data Interaction in AR and VR Using a Handheld Touch-Sensitive Device". In: *ISS'21: Interactive Surfaces and Spaces*. 2021 (cit. on pp. 8, 26, 30, 33).

[60] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum". In: Photonics for Industrial Applications. Ed. by Hari Das. Boston, MA, Dec. 21, 1995, pp. 282–292 (cit. on p. 1).

[61] Mark R Mine. "Virtual environment interaction techniques". *UNC Chapel Hill CS Dept* (1995) (cit. on pp. 1, 21, 22, 34).

[62] Erwan Normand and Michael J. McGuffin. "Enlarging a Smartphone with AR to Create a Handheld VESAD (Virtually Extended Screen-Aligned Display)". In: *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Oct. 2018, pp. 123–133 (cit. on pp. 14, 15, 36, 37).

[63] Alex Olwal and Feiner Steven. "The Flexible Pointer: An Interaction Technique for Selection in Augmented and Virtual Reality". In: *Proc. UIST*. Vol. 3. 2003, pp. 81–82 (cit. on pp. 21–23, 34).

[64] James R. Osborn and Alice M. Agogino. "An Interface for Interactive Spatial Reasoning and Visualization". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '92*. The SIGCHI Conference. Monterey, California, United States: ACM Press, 1992, pp. 75–82 (cit. on p. 23).

[65] Jeffrey S Pierce, Andrew S Forsberg, Matthew J Conway, Seung Hong, Robert C Zeleznik, and Mark R Mine. "Image plane interaction techniques in 3D immersive environments". In: *Proceedings of the 1997 symposium on Interactive 3D graphics*. 1997, 39–ff (cit. on pp. 22, 34).

[66] Patrick Reipschlager, Tamara Flemisch, and Raimund Dachselt. "Personal augmented reality for information visualization on large interactive displays". *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2020), pp. 1182–1192 (cit. on pp. 15, 36).

[67] Bernhard Reitinger, Alexander Bornik, Reinhard Beichel, Georg Werkgartner, and Erich Sorantin. "Tools for Augmented-Reality-Based Liver Resection Planning". In: Medical Imaging 2004. Ed. by Robert L. Galloway Jr. San Diego, CA, May 5, 2004, p. 88 (cit. on p. 34).

[68] Jun Rekimoto and Katashi Nagao. “The World through the Computer: Computer Augmented Interaction with Real World Environments”. In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology - UIST ’95*. The 8th Annual ACM Symposium. Pittsburgh, Pennsylvania, United States: ACM Press, 1995, pp. 29–36 (cit. on pp. 13, 14).

[69] Jie Ren, Yueting Weng, Chengchi Zhou, Chun Yu, and Yuanchun Shi. “Understanding Window Management Interactions in AR Headset + Smartphone Interface”. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI ’20: CHI Conference on Human Factors in Computing Systems. Honolulu HI USA: ACM, Apr. 25, 2020, pp. 1–8 (cit. on p. 16).

[70] Dieter Schmalstieg, L. Miguel Encarnação, and Zsolt Szalavári. “Using Transparent Props for Interaction with the Virtual Table”. In: *Proceedings of the 1999 Symposium on Interactive 3D Graphics - SI3D ’99*. The 1999 Symposium. Atlanta, Georgia, United States: ACM Press, 1999, pp. 147–153 (cit. on p. 14).

[71] Dieter Schmalstieg, Anton Fuhrmann, Gerd Hesina, Zsolt Szalavári, L. Miguel Encarnação, Michael Gervautz, and Werner Purgathofer. “The "Studierstube" Augmented Reality Project”. *Presence: Teleoperators and Virtual Environments* 11.1 (Feb. 2002), pp. 33–54 (cit. on pp. 3, 4, 34).

[72] Dieter Schmalstieg, Anton Fuhrmann, Zsolt Szalavari, and Michael Gervautz. “Studierstube-an environment for collaboration in augmented reality”. In: *CVE’96 Workshop Proceedings*. Vol. 19. 1996 (cit. on p. 3).

[73] Mickael Sereno, Lonni Besançon, and Tobias Isenberg. “Supporting Volumetric Data Visualization and Analysis by Combining Augmented Reality Visuals with Multi-Touch Input”. Version 021-023. *EuroVis 2019 - Posters* (2019), 3 pages (cit. on pp. 6, 26).

[74] Ken Shoemake. “Arcball Rotation Control.” *Graphics gems* 4 (1994), pp. 175–192 (cit. on p. 10).

[75] V Skala and A Brusi. “Two methods for iso-surface extraction from vlumetric data and their comparison”. *Machine Graphics and Vision* 9.1/2 (2000), pp. 149–166 (cit. on p. 29).

[76] Peng Song, Wooi Boon Goh, Chi-Wing Fu, Qiang Meng, and Pheng-Ann Heng. “WYSIWYF: Exploring and Annotating Volume Data with a Tangible Handheld Device”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11: CHI Conference on Human Factors in Computing Systems. Vancouver BC Canada: ACM, May 7, 2011, pp. 1333–1342 (cit. on pp. 6, 34).

[77] Jonathan Steuer. “Defining virtual reality: Dimensions determining telepresence”. *Journal of communication* 42.4 (1992), pp. 73–93 (cit. on p. 1).

[78] Hemant Bhaskar Surale, Aakar Gupta, Mark Hancock, and Daniel Vogel. “TabletInVR: Exploring the Design Space for Using a Multi-Touch Tablet in Virtual Reality”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI ’19: CHI Conference on Human Factors in Computing Systems. Glasgow Scotland Uk: ACM, May 2, 2019, pp. 1–13 (cit. on pp. 6, 7, 20, 26, 33, 34, 38).

[79] IE Sutherland. “The Ultimate Display. Information Processing Techniques Office”. In: ARPA, OSD, Proceedings of IFIP Congress. 1965 (cit. on p. 16).

[80] Zsolt Szalavari and Michael Gervautz. “The Personal Interaction Panel - a Two-Handed Interface for Augmented Reality”. *Computer Graphics Forum* 16.3 (Sept. 1997), pp. C335–C346 (cit. on pp. 3, 13, 14, 34).

[81] Z. Szalavári, D. Schmalstieg, A. Fuhrmann, and M. Gervautz. ““Studierstube”: An Environment for Collaboration in Augmented Reality”. *Virtual Reality* 3.1 (Mar. 1, 1998), pp. 37–48 (cit. on pp. 3, 4).

[82] Rui Tang, Long-Fei Ma, Zhi-Xia Rong, Mo-Dan Li, Jian-Ping Zeng, Xue-Dong Wang, Hong-En Liao, and Jia-Hong Dong. “Augmented Reality Technology for Preoperative Planning and Intraoperative Navigation during Hepatobiliary Surgery: A Review of Current Methods”. *Hepatobiliary & Pancreatic Diseases International* 17.2 (Apr. 2018), pp. 101–112 (cit. on pp. 1, 28).

[83] Dan Venolia. “Facile 3D Direct Manipulation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI ’93*. The SIGCHI Conference. Amsterdam, The Netherlands: ACM Press, 1993, pp. 31–36 (cit. on pp. 10–12).

[84] John Viega, Matthew J Conway, George Williams, and Randy Pausch. “3D magic lenses”. In: *Proceedings of the 9th annual ACM symposium on User interface software and technology*. 1996, pp. 51–58 (cit. on p. 24).

[85] Pierre Wellner, Wendy Mackay, and Rich Gold. “Back to the real world”. *Communications of the ACM* 36.7 (1993), pp. 24–26 (cit. on p. 13).

[86] Werner Wohlfahrter, L Miguel Encarnaçao, and Dieter Schmalstieg. *Interactive volume exploration on the study desk*. Citeseer, 2000 (cit. on pp. 3, 4, 29, 34).

[87] Luisa Wurm, Rubén García, Christoph Anthes, Dieter Kranzlmüller, and Wolfgang Höhl. “Benefits of Tablet Interfaces for Immersive Visualization in Information Visualization” (2016), p. 4 (cit. on pp. 17, 18).

[88] Hans Peter Wyss, Roland Blach, and Matthias Bues. “iSith-Intersection-based spatial interaction for two hands”. In: *3D User Interfaces (3DUI’06)*. IEEE. 2006, pp. 59–61 (cit. on pp. 21–23, 34).

[89] Yalong Yang, Tim Dwyer, Kim Marriott, Bernhard Jenny, and Sarah Goodwin. “Tilt Map: Interactive Transitions Between Choropleth Map, Prism Map and Bar Chart in Immersive Environments”. *IEEE Transactions on Visualization and Computer Graphics* 27.12 (2021), pp. 4507–4519 (cit. on p. 38).

[90] Stanislaw Zabramski. “Careless touch: A comparative evaluation of mouse, pen, and touch input in shape tracing task”. In: *Proceedings of the 23rd Australian Computer-Human Interaction Conference*. 2011, pp. 329–332 (cit. on p. 17).