

UNIVERSITY OF BREMEN

DOCTORAL THESIS

---

**Novel Algorithms and Methods for Immersive  
Telepresence and Collaborative VR**

**RGB-D Streaming, 3D Scene (Re-)Construction,  
and Avatar Visualization**

---

*A thesis submitted in fulfillment of the requirements  
for the degree of Doctor of Engineering (Dr.-Ing.)*

*to the*

Faculty of Computer Science

*Author:*  
Roland FISCHER

*Examiners:*  
Prof. Dr. Gabriel ZACHMANN  
Prof. Dr. Stefan MÜLLER

August 16, 2023

*Kolloquium:*  
October 05, 2023



## *Abstract*

Virtual reality (VR) is stated to be a key technology of the next decade with huge growth potential and is expected to transform many areas of business as well as daily life [73]. Thanks to technological advances in hard- and software, over the recent years, VR is getting used more frequently and moved from being solely a niche, experimental and expensive gadget to being a more widespread and mature (entertainment) device. Furthermore, it emerged as a serious and highly beneficial tool for various professional applications, which significantly enlarges the target group and user base. The application domains for VR are manifold and range from industry to entertainment and the scientific community. One concrete example is the virtual exploration of environments that are difficult or impossible to visit in person, e.g., in the field of archaeology and cultural heritage [53, 182]. Other examples are virtual 3D data/object inspection, e.g., in the medical, digital marketing, or architectural areas, and highly immersive and presence-inducing communication/telepresence, e.g., in healthcare or office/work settings. Additionally, VR is a natural fit for teaching and education tasks; either in industry settings, healthcare, or school. The main benefit of VR is that, with the help of innovative input and output devices, such as head-mounted displays, and stereoscopic vision, users can view, explore, and act in arbitrary computer-generated 3D environments in a more immersive and natural way than by looking at a simple 2D screen. Ideally, the users get a feeling of presence – as if they were actually there [152]. To that effect, a general goal is to make the environments and interaction metaphors as realistic and intuitive as possible, though this is not always a necessity.

The combination of VR and multi-user architectures promises to be especially beneficial, as it enables multiple (remote) people to enjoy the advantages of VR, interact with each other, and collaborate in the virtual 3D environment. For this reason, these systems are high in demand and, consequently, are developed increasingly often. However, there are still many challenges to overcome before these systems can truly gain traction and fully leave their still-lasting small-scale prototype character. Many of them are related to 3D visualization, reconstruction, or rendering of some kind. For instance, in collaborative VR, users are usually represented by 3D avatars. High-quality systems employ avatars based on live reconstructed point clouds that are captured using RGB-D cameras. Generating and rendering these high-quality avatars in real-time is a difficult and computationally demanding task. Moreover, streaming all the required data to all participants without generating high latencies or occupying vast amounts of bandwidth requires sophisticated compression algorithms. Other unsolved issues are, for example, how to combine immersive VR applications with high-detailed interactive 3D data visualization, such as computed tomography scans in medicine, or the suitable visualization of the popular teleportation locomotion in multi-user settings, which can be very confusing otherwise. However, to develop and employ immersive, effective multi-user VR applications, it is not enough to consider how to visualize confined live scenes, avatars, and selected 3D data, but also the virtual environment itself. Most often, the environment consists of manually created meshes but in some use cases, such as virtual testbeds, there is a need for having (sometimes even multiple) vast, plausible-looking 3D landscapes. Generating these is an often overlooked topic that is not trivial, though. These environments have to be created procedurally in order to keep the workload in check, which in turn requires smart algorithms to quickly compute realistic, feature-rich terrains.

With this work, we address the abovementioned challenges (e.g., low-latency data compression and streaming, real-time avatar reconstruction and rendering, etc.) and focus on the question of how novel 3D rendering and visualization methods can improve multi-user VR applications. A core contribution of the thesis is, therefore, the design, development, and evaluation of a low-latency, real-time point cloud streaming and rendering pipeline for VR-based telepresence. With this, we are able to depict high-quality live-captured 3D scenes and avatars in shared virtual environments. Telepresence applications such as ours often rely on RGB-D cameras. As the captured depth data is inherently incomplete and still takes up a huge amount of space, we propose novel methods for RGB-D/depth image enhancement and compression, which are crucial tasks for using and streaming data in multi-user applications that employ RGB-D cameras. Related to telepresence, we also tackle the aforementioned issue of the potentially confusing teleport locomotion in multi-user environments, where the avatar is simply placed at the target destination. We do this by proposing and evaluating multiple 3D visualizations, intended to prevent confusion and preserve the observer's feeling of presence. Together, these contributions allow for an immersive real-time 3D visualization of remote physical scenes and avatars in collaborative VR environments. After this, we also considered the 3D environment itself and tackled the issue of the labor-intensive generation of large and detailed virtual environments, which are required for some application domains. We designed and developed multiple methods to procedurally generate realistically-looking terrains for VR applications. These are specifically capable of creating plausible biome distributions as well as natural water bodies. Multi-user VR applications can include other elements, too, though. For instance, in the medical area, volumetric data such as computed tomography scans. Hence, another question was how to visualize this data in immersive multi-user VR environments for collaborative inspection. Our solution to this is a combination of a custom direct volume rendering solution with an Unreal Engine 4-based collaborative VR application, which allows for interactive, high-quality volumetric visualizations. Lastly, we found that teaching and education are important areas for multi-user VR, however, it is not adequately explored yet, if collaborative learning in VR, e.g., anatomy learning, is as beneficial as it is classically. Thus, we conducted a study and investigated the effects of collaborative anatomy learning in VR using a collaborative VR anatomy atlas that provides accurate interactive 3D models of all the human organs and anatomical structures. With this, we considered all visualization-related challenges of multi-user VR, from live-captured scenes and avatars to selected 3D data, and to the virtual 3D environments itself, and therefore help to elevate multi-user VR to another level.

## *Acknowledgements*

I am very grateful to my supervisor, Prof. Dr. Gabriel Zachmann, for giving me the opportunity to pursue my thesis in a fantastic research environment and for his continuous support and excellent guidance. Whenever needed, he always gave me precious advice and helpful feedback.

I also would like to express my gratitude to Prof. Dr. Stefan Müller for accepting the co-advisorship.

Moreover, I wish to thank all my co-authors as well as academic and industrial collaborators for the fruitful cooperative work without which this thesis undoubtedly would not exist. Namely, Dr. René Weller, Philipp Dittmann, Andre Mühlbrock, Christoph Schröder-Dering, Janis Rosskamp, Thomas Hudcovic, Dr. Verena Uslar, Prof. Dr. Dirk Weyhe, Marc Jochens, Farin Kulapichitr, Judith Boeckers, Kai-Ching Chang, and Anton Schlegel.

Naturally, I would also like to thank all my colleagues from the CGVR research group, especially, my office mate Hermann Meißenhelther. I always enjoyed the very friendly and productive atmosphere and the insightful discussions. Almost all of them contributed in some form to shape this thesis the way it is today.

Thanks go also to all my students and study participants for their work and efforts.

Moreover, there is no doubt that I am greatly indebted to my mother who supported me all the time throughout this journey and made my life so much easier. She was always very understanding and had an ear for me and my problems.

I also would like to express my sincere gratitude to Haya for her support. She always gave me confidence, encouraged me to keep going and do my best, and offered her help whenever possible.

Last but not least, I cannot thank my good friends Niklas, Mario, and Alex enough. They were always there for me with helpful advice and lifted my spirits when needed. Also, by helping with the proofreading, they made sure that at least some parts are correct.



# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation	1
1.2 Overview and Contributions	6
<b>2 Overview and State of the Art of the Area of Multi-User VR</b>	<b>11</b>
2.1 Terminology	11
2.1.1 Virtual Reality	11
2.1.2 Presence, Immersion, and Embodiment	13
2.2 Important Aspects of (Multi-User) VR	15
2.2.1 Interaction and Locomotion	16
2.2.2 Avatars	18
2.2.3 Sensing and Tracking Methods	18
RGB-D/Depth Enhancement	20
2.2.4 Reconstruction and Rendering Methods	21
2.2.5 Streaming and Compression of Sensor Data	23
2.2.6 Virtual Environments	23
2.3 Example Application Domains	25
<b>3 Algorithms and Architectures for Telepresence in Multi-User VR</b>	<b>31</b>
3.1 Development and Evaluation of a Point Cloud Streaming and Rendering Pipeline	31
3.1.1 Introduction	32
3.1.2 Related Work	33
3.1.3 System Overview	36
Multi-User VR Environment	37
Point Cloud Streaming	37
Point Cloud Registration and Rendering	39
3.1.4 Results	42
Performance	42
Network	45
3.1.5 User Studies	46
Study 1: Qualitative Feedback, Presence, and Preference	46
Study 2: Comparison of Point Cloud Rendering Solutions	48
3.1.6 Limitations	50
Face Reconstruction	50
3.1.7 Conclusions and Future Work	52
3.2 (Improved) Lossless Depth Image Compression Methods	53
3.2.1 Introduction	54
3.2.2 Related Work	55

3.2.3	Proposed Approach	56
	Adaptive Span-Based Prediction	57
	Inter-Frame Delta Computation	58
	Further bit reduction	59
	Parallel Execution	59
3.2.4	Results	60
	Additional Data	66
	Evaluation of Lossy Video Compression	66
3.2.5	Conclusions and Future Work	72
3.3	Enhancement of Depth Images using Deep Neural Networks	74
3.3.1	Introduction	75
3.3.2	Related Work	76
3.3.3	Categorization of Depth Errors	77
3.3.4	Proposed Approach	79
	Datasets	79
	Preprocessing Pipeline	79
	Network Details	83
	Training Procedure	85
3.3.5	Results	86
3.3.6	Conclusions and Future Work	96
<b>4</b>	<b>Perception of Teleport Visualizations in Multi User VR</b>	<b>99</b>
4.1	Introduction	99
4.2	Related Work	101
4.3	Proposed Teleport Visualizations	103
4.4	Study	104
4.4.1	Hypotheses	104
4.4.2	Experimental Setup	106
4.4.3	Procedure	108
4.5	Results	110
4.5.1	Participants	110
4.5.2	Qualitative and Quantitative Data	110
4.6	Discussion	118
4.6.1	Comparison of IFoV and OFoV Scenarios	123
4.6.2	Comparison of Slow and Fast Experiment	123
4.7	Limitations	124
4.8	Conclusions and Future Work	125
<b>5</b>	<b>Large-scale Procedural Terrain Generation for VR Environments</b>	<b>127</b>
5.1	AutoBiomes: Procedural Generation of Multi-Biome Landscapes	127
5.1.1	Introduction	128
5.1.2	Related Work	129
5.1.3	Proposed Approach	130
	Base Terrain	131
	Climate Simulation	133
	Terrain Refinement	137
	Asset Placement	138
5.1.4	Results	140
5.1.5	Conclusions and Future Work	146
5.2	Procedural Generation of Landscapes with Water Bodies Using Artificial Drainage Basins	146



5.2.1	Introduction	147
5.2.2	Related Work	147
5.2.3	Overview of our Approach	149
5.2.4	Our Terrain Generation Pipeline in Detail	150
	Ocean Borders	150
	Regions	152
	River Networks and Lakes	154
	Terrain	158
	Visualization	159
5.2.5	Results	160
	Complexity Analysis	160
	Performance Evaluation	161
	Qualitative Evaluation	162
5.2.6	Conclusions and Future Work	163
5.3	Procedural Terrain Lookalikes - Generating Extraterrestrial Planetary Surfaces for VR Testbeds	166
5.3.1	Introduction	167
5.3.2	Related Work and Promising Approaches	167
	Heuristic- and Noise-based Lookalikes	168
	Machine/Deep Learning-based Lookalikes	169
<b>6</b>	<b>Applications for Multi-User VR in the Medical Field</b>	<b>173</b>
6.1	Volumetric CT Data Visualization for Collaborative VR Environments	173
6.1.1	Introduction	174
6.1.2	Related Work	175
6.1.3	Proposed Approach	176
	Direct Volume Rendering	177
	Collaborative VR	180
6.1.4	Results	183
6.1.5	Conclusions and Future Work	193
6.2	Anatomy Learning through a Collaborative VR Anatomy Atlas	193
6.2.1	Introduction	194
6.2.2	Related Work	194
6.2.3	Implementation	195
6.2.4	Study	195
6.2.5	Design and Setup	196
	Procedure	196
6.2.6	Results	197
6.2.7	Discussion	198
6.2.8	Conclusions and Future Work	198
<b>7</b>	<b>Conclusions and Outlook</b>	<b>201</b>
7.1	Summary	201
7.2	Outlook	204
<b>A</b>	<b>Publications</b>	<b>207</b>
<b>B</b>	<b>Additional Plots for Section 3.2</b>	<b>209</b>
	<b>Bibliography</b>	<b>215</b>



# List of Abbreviations

<b>3DMM</b>	<b>3 Dimensional Morphable Model</b>
<b>AR</b>	<b>Augmented Reality</b>
<b>CAVE</b>	<b>Cave Automatic Virtual Environment</b>
<b>CNN</b>	<b>Convolutional Neural Network</b>
<b>CRF</b>	<b>Constant Rate Factor</b>
<b>CT</b>	<b>Computed Tomography</b>
<b>DEM</b>	<b>Digital Elevation Model</b>
<b>DVR</b>	<b>Direct Volume Rendering</b>
<b>FOV</b>	<b>Field Of View</b>
<b>GAN</b>	<b>Generative Adversarial Network</b>
<b>GOP</b>	<b>Group Of Pictures</b>
<b>GUI</b>	<b>Graphical User Interface</b>
<b>HMD</b>	<b>Head-Mounted Display</b>
<b>HU</b>	<b>Houndsfield Unit</b>
<b>ICU</b>	<b>Intensive Care Unit</b>
<b>IMU</b>	<b>Inertial Measurement Unit</b>
<b>IPQ</b>	<b>Igroup Presence Questionnaire</b>
<b>IR</b>	<b>Infrared</b>
<b>LAO</b>	<b>Local Ambient Occlusion</b>
<b>LOD</b>	<b>Level Of Detail</b>
<b>MR</b>	<b>Mixed Reality</b>
<b>MRI</b>	<b>Magnetic Resonance Imaging</b>
<b>OCN</b>	<b>Optimal Channel Network</b>
<b>OR</b>	<b>Operating Room</b>
<b>PSNR</b>	<b>Peak Signal-To-Noise Ratio</b>
<b>ROI</b>	<b>Region Of Interest</b>
<b>PTG</b>	<b>Procedural Terrain Generation</b>
<b>SDK</b>	<b>Software Development Kit</b>
<b>SLAM</b>	<b>Simultaneous Localization And Mapping</b>
<b>SSAA</b>	<b>Super Sampling Anti Aliasing</b>
<b>SUS</b>	<b>System Usability Scale</b>
<b>TOF</b>	<b>Time Of Flight</b>
<b>UE</b>	<b>Unreal Engine</b>
<b>UI</b>	<b>User Interface</b>
<b>VR</b>	<b>Virtual Reality</b>
<b>VTK</b>	<b>Visualization Toolkit</b>



## Chapter 1

# Introduction

With this chapter, we would like to introduce the topic of this thesis and its wider context. First, in Section 1.1, we motivate the thesis by talking about (multi-user) virtual reality, why it is getting increasingly popular, its importance and application domains, but also its current issues. Then, in Section 1.2, we will give an overview of our contributions with which we tackle these issues.

### 1.1 Motivation

Virtual reality (VR) is a key technology of the next decade with huge growth potential and is expected to transform many areas of business as well as daily life [73]. For instance, a market research report from Fortune Business Insights [154] states that “the healthcare industry is expected to witness significant disruption across the industry with VR applications”. According to forecasts by Statista [271], the combined augmented reality (AR), VR, and mixed reality (MR) market size is projected to multiply 9-fold from 28 billion dollars in 2021 to 252 billion in 2028. Examining the IDC worldwide augmented and virtual reality spending guide [150], worldwide spending on AR and VR is projected to increase from 14 billion dollars in 2022 to 51 billion in 2026, of which VR will account for more than 70 percent. Thanks to technological advances in hard- and software, over the recent years, VR is getting used more frequently and moved from being solely a niche and expensive gadget for gaming to being a more widespread and mature entertainment device. Furthermore, it emerged as a serious and highly beneficial tool for various professional applications, which makes it popular with a broader audience [51].

The application domains for VR are manifold and range from industry to entertainment and the scientific community. For instance, in the fields of cultural heritage, tourism, and archaeology, VR, in conjunction with modern scanning techniques, can be utilized to virtually reproduce and then explore environments that are difficult or impossible to visit in person [182]. Figure 1.1 (top left) depicts how this process, from data recording to visualization in VR, could look like. Similarly, VR also can be employed for virtual 3D data/object inspection and exploration, e.g., in the medical [153, 296], digital marketing/retail/fashion [424, 418], or architectural areas. One example of this is depicted in Fig. 1.1 (bottom left), in which VR and volume rendering technology are combined to 3D visualize and interactively inspect anatomical data. Moreover, highly immersive and presence-inducing VR-based communication/telepresence systems have great potential and are eagerly tested in many domains such as healthcare [210], or general professional work settings [88]. For instance, Fig. 1.1 (bottom right) shows a point cloud-based VR telepresence system designed to replace standard video conferencing. Additionally, VR is a natural fit for teaching and education tasks; either in industry settings [262], healthcare [172], or school. Figure 1.1 (bottom mid) shows an example of a VR dental simulator for

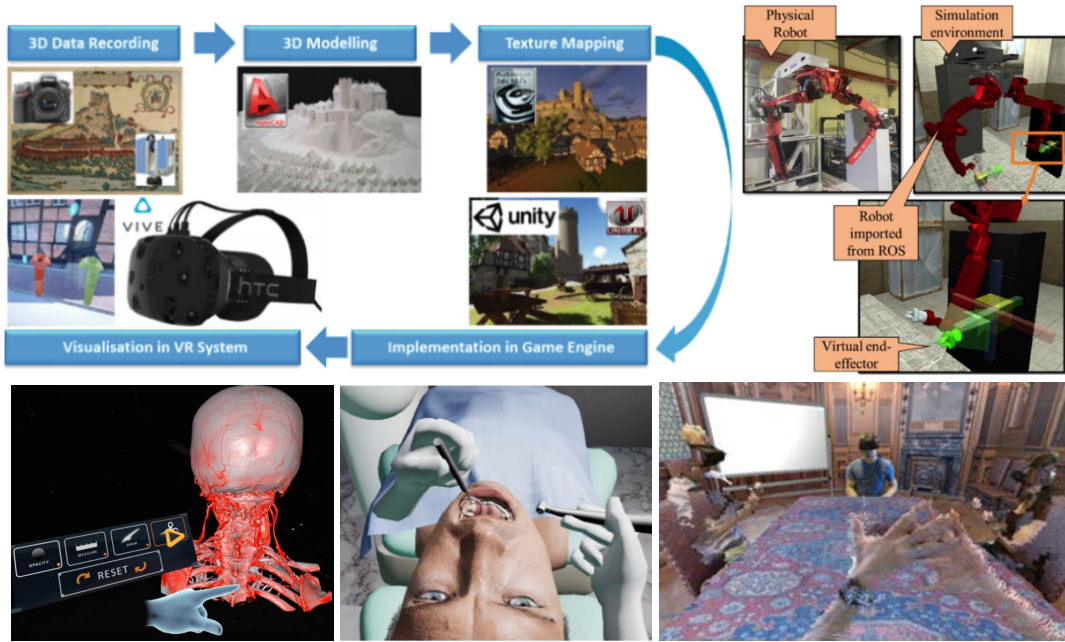


FIGURE 1.1: Application domains of (collaborative) VR. Top left: Digital twin generation workflow in cultural heritage [182]. Top right: VR Simulation environment for industrial robot control [382]. Bottom Left: 3D data visualization in medicine [153]. Bottom mid: Shared VR environment for dental surgical training [172]. Bottom right: Telepresence application for VR conferencing [88]

teaching and training dental surgeons. Generally, VR environments lend themselves to getting used as simulation environments, e.g., for autonomous driving [426], in medicine [280], or in industrial settings [382]. See for example Fig. 1.1 (top right), which depicts a VR simulation environment for industrial robot control. With this, all actions and movements can be virtually simulated and validated before being executed physically. The commercial use cases that are forecast to receive the largest investments are training and collaboration [150].

The main benefit of VR is that users can view, explore, and act in arbitrary computer-generated 3D environments in a more immersive and natural way than by looking at a simple 2D screen. This is achieved with the help of innovative input and output devices. For instance, head-mounted displays (HMD) provide stereo vision while pairs of tracked controllers provide haptic feedback. Ideally, the users get a feeling of presence – as if they were actually there [319]. To that effect, a general goal is to make the environments and interaction metaphors as realistic and intuitive as possible, though this is not always a necessity. Although VR got a lot of attention lately, the concept itself is not particularly new. Sutherland [366] described something akin to today’s concept of VR already in 1965 when he wrote about a display that provides a computer-controlled environment in which everything is experienced as if it would be real. Following that, the first VR devices were developed in the seventies and eighties, though these systems were rather crude back then. The first commercial products for consumers were released in the nineties, however, the lacking technology was still limiting its wider success [337]. The devices were too heavy, cumbersome, and expensive. Also, the visual quality was lacking, both from a hard- and software perspective (e.g., low screen resolution, simple rendering methods). Only in the last decade, with improved HMDs and VR systems such as the Oculus Rift, HTC Vive, and PlayStation VR, we saw a more serious and widespread

interest in VR leading to a significant expansion of the VR industry and a new wave of VR application development [68].

The combination of VR and multi-user architectures promises to be especially beneficial, as it enables multiple (remote) people to enjoy the advantages of VR, interact with each other, and collaborate in the virtual 3D environment [63]. For this reason, these systems are high in demand and, consequently, are developed increasingly often. Closely collaborating with colleagues or other people, in general, is highly important and beneficial for work [272], studying [195], or playing. Ideally, people would be and work together in the same physical location, however, that is not always feasible. For instance, in today's highly connected and specialized world, teams often consist of people and experts from different cities or even continents that cannot constantly travel or relocate. Also, considering the recent covid-19 pandemic, there is always the possibility of extraordinary circumstances that make physical meetings impossible. Moreover, it highlighted the importance of working-from-home modalities for employees. Most software systems and communication tools that get used today to enable remote collaboration are just classic videoconferencing tools or rather simple telepresence systems, though. They are highly limited regarding immersive visualization and natural interaction, both with the system itself as well as the other people. An actual strong feeling of (tele)presence cannot be achieved with these systems and productive teamwork is hampered [43].

In the following, we will consider some concrete example domains where novel collaborative VR systems could be employed beneficially to improve the current workflows and look at the advantages they could provide. The first example we want to highlight is remote surgery assistance. Nowadays, surgeries can be incredibly complex and require the involvement of multiple doctors from various disciplines, such as surgeons, anesthetists, and radiologists. This makes surgery inherently a collaborative task. Naturally, nurses/assistants are present in the operating room (OR), too. In many cases, junior doctors and medical students join in as well, since observation of actual surgeries is a major avenue of education [23]. One issue is the inhomogeneously distributed knowledge of the doctors, though [95]. In some cases, the assistance of specialized experts or more experienced doctors is needed. However, they are usually not on site and have to travel, which takes valuable time, or they cannot visit at all. Today, the prevalent solution for these situations is to receive remote assistance via phone or videoconferencing. This is not ideal as the remote expert cannot get a comprehensive visual overview of the situation. A multi-user VR telepresence system with an array of RGB-D cameras mounted in the operating room, such as the one by Roth et al. [311], could provide remote experts with high-fidelity 3D live visuals of the scene. Especially having and observing the scene in 3D would be very helpful as it provides crucial depth cues and helps to have a better understanding of the spatial relations between objects and organs. Moreover, remote and local doctors would have advanced and intuitive options to interact with each other, especially when represented by real-time avatars. How such a system could look like is depicted in Fig. 1.2. The left image shows the live-captured operating room with the local doctor and two avatars representing remote experts. The top right picture shows the remote expert using a VR system for tracking and interaction and the picture on the bottom right is his/her first-person view of the operating room. An additional advantage of such systems would be that the medical students could observe and learn from remote locations, which would be more convenient and free up space in the OR [79]. Naturally, such VR telepresence systems could also be employed to help doctors to teach virtually or visualize and discuss 3D patient data; again, benefiting from the 3D representations and depth cues [23].



FIGURE 1.2: Mixed reality teleconsultation system for intensive care units (ICU) by Roth et al. [311]. Left: the ICU with a local doctor using MR glasses and two remote experts for consultation that are represented by 3D avatars. Top right: a remote expert using VR to consult from a distance. Bottom right: The remote expert's view of the local scene.

Another example use case for collaborative VR from a completely different area that we would like to present is virtual testbeds. A virtual testbed is a software system that is designed to allow the simulation and testing of hard and software systems in a virtual environment. Usually, it is done before actual real-world tests are conducted. Virtual testbeds are valuable tools for researchers and engineers, especially in earlier project phases, as they allow for quicker and more cost-effective testing compared to conducting physical tests and, at the same time, can serve as a communication platform. Another advantage is that they are not restricted physically, and therefore, can simulate arbitrary conditions which may not be available in the real world. Also, parameters can be changed on the fly to alter the test conditions quickly. Example areas in which virtual testbeds get employed are, among others, the automotive industry, robotics, and space exploration [309]. There, they simulate, for instance, communication interfaces, sensor input, and the physical terrain to which the car or robot then reacts. Doing this in a virtual environment allows for the testing of software components and fixing initial flaws before costly real-world field tests have to be conducted. This is especially important for space applications and planetary exploration [407]. Properly and easily monitoring the virtual tests is often problematic, though, because of limited usability and lack of visual feedback. However, observing the produced data and the system's behavior is highly important. Embedding the virtual testbed in a multi-user VR application would help to provide immersive high-fidelity visual feedback during the testing, i.e., 3D visualizations of sensor data, as well as natural interaction possibilities for one or multiple engineers or researchers. This would allow them to dynamically interact with the system, collaboratively discuss the progress and behavior, and eventually help with complex decision-making and planning. An example is depicted in Fig. 1.3. The left picture shows how two users use the VR system to interact in the shared virtual testbed and the right picture depicts the autonomous vehicles that get simulated as well as a color-coded visualization of sensor data.

However, there are still many challenges to overcome before multi-user VR systems can truly gain traction and fully leave the small-scale prototype character that



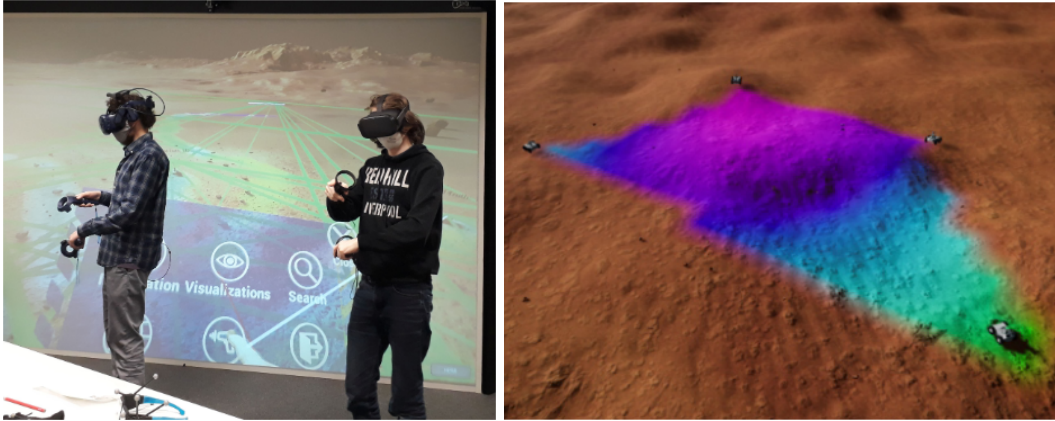


FIGURE 1.3: Collaborative VR testbed for Mars exploration by Weller et al. [407]. Left: two users explore and interact with the shared virtual environment using VR headsets and controllers. Right: simulated autonomous vehicles exploring the virtual terrain and visualization of sensor data.

they often still pertain. Many of them are related to 3D visualization, reconstruction, and rendering of some kind, or related tasks such as compression and streaming of the data. For instance, in collaborative VR, users are usually represented by 3D avatars. To produce a strong feeling of presence for the users, both spatially and socially, and convey the sensation of embodying the depicted avatar in the virtual world, the avatars have to be highly detailed, and most importantly personalized [392]. Just capturing images of the users beforehand and then applying them on a mesh-based model is not enough though. Mimics and dynamic changes in general appearance should be depicted faithfully in real-time too, in order to get truly immersive and accurate depictions of the users and their emotions in multi-user scenarios [416]. Moreover, it was shown that a high degree of freedom regarding the avatar's motion is important as well. As we see, having suitable avatars that accurately represent the visuals and movements of the users in real time is not trivial. High-quality systems often employ avatars based on live reconstructed point clouds that are captured using RGB-D cameras. RGB-D cameras capture not only the color for each pixel but also the depth to the objects, which helps to reconstruct 3D scenes, and in this case, 3D avatars. Generating and rendering these high-quality avatars in real-time is a difficult and computationally demanding task [90]. There are various approaches to reconstructing and rendering 3D avatars from raw RGB-D data, e.g., point clouds, mesh reconstruction, and implicit surfaces combined with ray casting. All of these are demanding tasks to be done in real time when also aiming for high-quality results.

Additionally, raw RGB-D data is prone to noise and artifacts which have to be dealt with in preprocessing steps in order to get smooth results [381]. Denoising and inpainting algorithms explicitly designed for depth data and real-time performance are sparse though, and still leave much room for improvement. For instance, one of the few works regarding depth completion (without color guidance) is the work by Jin et al. [163], however, only small holes in the depth images are handled well. Having multiple RGB-D cameras and combining the results at least helps in preventing occlusions, on the other hand, however, the cameras then have to be registered to each other precisely at all times.

An imperative aspect of collaborative VR is streaming all the required data to all

participants without generating high latencies or occupying vast amounts of bandwidth. Especially RGB-D data has a large size when send raw, which makes sophisticated compression algorithms necessary. The color data can easily be compressed with current video compression algorithms, such as H.264, but for the depth data, custom algorithms are needed, as the characteristics of the data are inherently different [412].

Having high-quality avatars in collaborative VR naturally raises also the question of how to move throughout the virtual environment. Small-scale distances can be covered using direct tracking but for larger distances, other locomotion metaphors are needed. One of the most popular ones is the teleport metaphor as it is easy and not prone to induce cybersickness. However, a currently unsolved issue is how to suitably visualize this inherently non-continuous process to onlookers in multi-user settings. Just disappearing and reappearing without any motion cues can be very confusing and reduce the crucial feeling of presence [133].

Another rendering-related question is how other spatial/3D data, such as computed tomography (CT) scans, which are a vital tool used in medicine, can be rendered in immersive collaborative VR applications so that doctors can inspect and interact with them together. Normally, if employed at all, such data gets visualized with specialized programs that are viewed individually on 2D screens and interacted with using classic user interfaces, while the game engines, which (collaborative) VR applications are based on and that provide 3D visuals, don't support the specialized rendering algorithms for such data, e.g., volume rendering techniques.

As discussed, to develop and employ immersive, effective multi-user VR applications, it is important to visualize confined live scenes, avatars and selected 3D data. However, we also need to consider the virtual environment itself. Most often, the environment consists of manually created meshes but in some use cases, such as the aforementioned virtual testbeds, there is a need for having (sometimes even multiple) vast, plausible-looking 3D landscapes to test robots, cars, or other software systems on [69]. Generating these landscapes is an often overlooked topic that is not trivial though. These environments have to be created procedurally in order to keep the workload in check, which in turn requires clever algorithms to quickly compute realistic, feature-rich terrains that are also easily controllable and usable for the designer [119].

As can be seen, there are various challenges with multi-user VR that require attention. How we tackle these through our thesis and the individual contributions, will be discussed in the following section.

## 1.2 Overview and Contributions

To summarize, this thesis aims at addressing the aforementioned challenges that are currently present in multi-user VR, thereby elevating the state of the art to a new level, and eventually improving the quality and feasibility of multi-user VR applications in practice. To achieve this, we present a series of novel contributions spread across various important subtopics. Figure 1.4 gives an overview of the challenges (depicted in blue) and our corresponding solutions/contributions (depicted in green).

Before actually presenting the contributions of this thesis, Chapter 2 will introduce the current state of the art in the field of multi-user VR and briefly touch up on a couple of relevant topics and underlying background information that might be helpful for the understanding of the contributions.

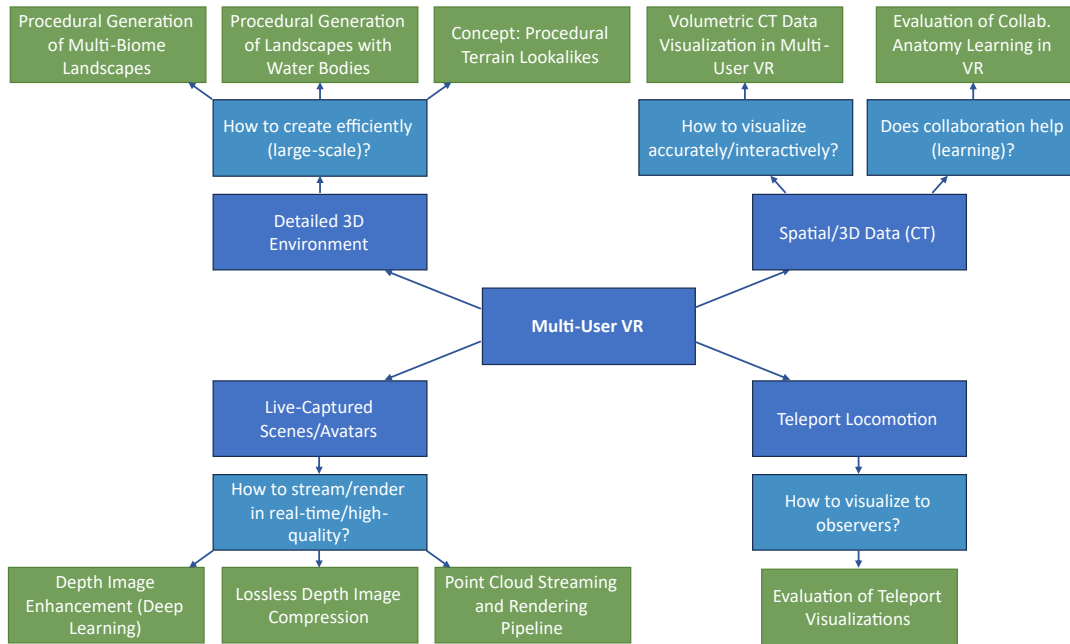


FIGURE 1.4: Overview of the challenges of multi-user VR (blue) and our corresponding solutions/contributions (green).

Then, Chapter 3 deals with the first challenges of multi-user VR, concretely, challenges regarding telepresence in VR, such as real-time streaming and 3D reconstruction. The first contribution in this chapter (presented in Section 3.1) is the design, development, and evaluation of a point cloud rendering and streaming system that addresses a central aspect of VR telepresence applications: the high-quality real-time 3D visualization of physical scenes and remote people in a shared virtual environment, as well as, the transmission of the data to and between all the distributed participants. As described above, having detailed, personalized avatars reconstructed and rendered in real-time is highly challenging and the arising data loads pose an issue, too. Our proposed telepresence system combines an immersive multi-user VR system with a real-time RGB-D streaming pipeline and two fast, custom point cloud rendering solutions integrated into a state-of-the-art 3D game engine. As an example application domain, and to show the practical viability of our system, we chose the medical field, more precisely, remote surgery assistance. Here, our solution enables remote experts to meet in a multi-user virtual operating room, view live-streamed and 3D-visualized operations, interact with each other, and collaboratively explore medical data. Our telepresence system can be easily applied in other fields and scenarios though. Our streaming pipeline is designed to be all-encompassing and prioritize low latencies. It is capable of handling multiple (Azure Kinect) RGB-D cameras per location, as it includes efficient real-time compression and filtering algorithms. Moreover, our system is easy to set up and includes the registration between all the sensors and the VR system. As for the rendering of the point clouds, we chose to develop a splatting method as well as a competing fast mesh-based solution. The participants are, too, visualized using live-captured and rendered point clouds. To handle the reconstruction of the users' faces, which are obstructed by the HMDs, we also developed a prototype based on eye- and face tracking, 3D Morphable Face Models (3DMM), as well as machine-learning-based mimics classification and interpolation method. To evaluate our proposed system, we conducted two user studies

with doctors and medical students in which we explored possible use cases, compared the different rendering solutions, and investigated the system regarding aspects such as spatial and social presence, realism as well as preference.

After this, we shift our attention to the specific task of real-time processing of RGB-D data. As remote scenes and personalized avatars usually are visualized using RGB-D sensors, and it is not feasible to transmit and use the data as-is – we did talk about the issue of noise and artifacts earlier –, processing tasks such as compression, denoising, and inpainting of missing areas are crucial and virtually necessary aspects of telepresence and multi-user VR applications (such as ours). Thus, our second contribution (in Section 3.2) is designed to further reduce the huge data loads that have to be streamed when using RGB-D cameras for avatar/ scene capture. As explained earlier, especially the depth images take up a lot of space and are hard to compress. Hence, we present a real-time, lossless depth image compression algorithm that is based on the RVL algorithm but achieves a significantly higher compression ratio. We achieve this by an improved span-wise adaptive intra-image prediction, the addition of a final entropy-coder stage, and an additional inter-frame delta computation. To speed up the computations, our algorithm is implemented using multi-threading. Finally, we did extensive experiments comparing the effectiveness of our proposed algorithm with various other lossless depth compression algorithms for different RGB-D cameras and a brief evaluation of lossy video compression algorithms applied to depth data.

Section 3.3 then presents another contribution directed at RGB-D processing, namely, an approach to quickly in-paint and enhance depth images, which usually contain various holes and areas with missing data that degrade the 3D reconstruction quality. Our proposed method is to utilize existing deep-learning models that were originally designed for color image inpainting. Concretely, we chose two promising U-Net-based network models: the first one uses partial convolutions, while the second one is based on a generative adversarial network (GAN) architecture. Both of our models, in contrast to many others, don't need any color images for guidance. For comparison, we also adopted a basic U-Net and the more sophisticated LaMa network. To evaluate our models, we did conduct a detailed quantitative and qualitative evaluation using two public and a custom self-recorded dataset.

Chapter 4 is concerned with another important aspect affecting telepresence in VR: the issue of suitably visualizing the popular teleport locomotion metaphor to other users in shared environments. We discussed that the lack of any motion cues can be confusing for observers. Our contribution is, therefore, the design, implementation, and evaluation of multiple 3D visualizations intended to prevent confusion and preserve the onlooker's feeling of presence. From the vast design space of possible visualizations, we chose to include continuous ones as well as discontinuous ones, including particle and portal effects, a quick dash, and a full walking animation. In a subsequent study, we investigated the visualizations' effects on the user experience and the spatial as well as social presence as perceived by the observers. Additionally, we compared how different paces affect the perception of the visualization.

Next, Chapter 5 deals with the effective generation of large-scale 3D environments, which are required in some (multi-user) VR applications. For instance, environmental simulation environments or virtual testbeds that simulate the navigation and interaction of unmanned vehicles and robots with different terrains. We discussed that generating these large detailed terrains manually is not a feasible option. Thus, in Section 5.1, we present a procedural terrain generation (PTG) system that is capable of efficiently creating vast terrains with plausible biome distributions

and therefore different spatial characteristics. This is done by combining several synthetic procedural terrain generation techniques with digital elevation models (DEMs) and a simplified climate simulation. A major focus was to generate realistic terrains while keeping simultaneously computation times low and still considering usability and flexibility. As part of our system, we also propose an effective and easy-to-use biome- and rule-based local-to-global model to populate the terrain with assets that follow complex multi-object distributions.

An often overlooked aspect of procedural terrain generation is the generation of landscapes that feature plausible water bodies. Therefore, in Section 5.2, we present a method and pipeline for the quick and easy procedural generation of large, plausible-looking landscapes which include and integrate believable water bodies, i.e., river networks and lakes. We achieve this by an approach inverse to the usual way: we first generate rivers and lakes based on artificial drainage basins and then create the actual terrain by “growing” it, starting at the water bodies. In order to demonstrate our proposed approach, we have developed a prototype application in Unity. In this prototype, we have applied a pipeline approach that makes it easy to evaluate intermediate results, emphasizes a workflow with quick iterations, and balances user control and automation. That means the first stages provide great control over the layout of the landscape while the later stages take care of the details with a high degree of automation.

Section 5.3 is then concerned with the procedural generation of terrain lookalikes to be used in virtual testbeds and simulation environments. Specifically, using space mission simulation as an example application domain, we consider the generation of extraterrestrial planetary surfaces (such as the one on Mars) based on input DEMs in order to quickly produce many slightly varying terrain lookalikes. This contribution is designed as more of an outlook into where and how PTG systems could be beneficially employed and to show the vast range of application domains and is thus strictly theoretical.

After this, in Chapter 6, we show two example applications from the medical area where multi-user VR can be beneficially employed. The first one is the suitable, accurate visualization of and dynamic interaction with spatial/3D data, i.e., medical data such as CT scans, in collaborative environments. Our contribution to this, presented in Section 6.1, is an easy-to-use and expandable system for volumetric CT data visualization with support for multi-user VR interactions. For this, we combined an immersive multi-user application based on the Unreal Engine (UE) 4 with a custom direct volume renderer. Real-time performance and good visualization quality are achieved by the implementation of several optimization and lighting techniques. Additionally, our system is capable of visualizing multiple windows in parallel in an easy and effective manner, thanks to a custom pipeline for processing the CT images.

This is followed up by a multi-user anatomy learning application and an accompanying user study, that we present in Section 6.2. The application allows multiple users to freely explore the virtual environment and an anatomical model of a human that consists of individual, interactive organs and structures. The focus of this work lay in investigating the effectiveness of collaborative anatomy learning in VR (in contrast to individual learning). Therefore, we conducted a user study to investigate the learning progress as well as the usability for both individual and multi-user learning. Moreover, this contribution serves as an example of how the interactive 3D visualization of visual data in multi-user VR can be successfully applied and beneficially employed.

Finally, Chapter 7 draws an overall conclusion of all the covered topics and the thesis in general. Also, a brief outlook is given on which areas might be in need of further research and what could be viable approaches.

At this point, we want to highlight the fact that a number of the contributions were published in a similar form in the publications listed in Appendix A.

After this brief overview of all the chapters and contributions that are to come, we will follow up with an introduction of the state of the art of multi-user VR and its wider context.

## Chapter 2

# Overview and State of the Art of the Area of Multi-User VR

This chapter gives a brief overview of the wide and complex area of multi-user VR and closely related subjects such as avatars, presence, and simulation environments. Moreover, it provides the necessary background knowledge about multi-user VR, commonly used (rendering) techniques of the area, and the current state of the art. It is intended to lay the theoretical foundations for the thesis contributions in the later chapters, help to frame them in a wider context, and supplement the more specific information given in the respective related work sections. To do so, we will first discuss the relevant terminology in Section 2.1, then talk about important and necessary aspects of (multi-user) VR systems in Section 2.2, and lastly present the diverse application domains, including a range of concrete applications, in Section 2.3.

### 2.1 Terminology

In order to get a common understanding of the concept of VR (Sec. 2.1.1) and related terms such as presence, and embodiment (Sec. 2.1.2), we will first discuss the terminology of these concepts in this Section.

#### 2.1.1 Virtual Reality

The concept of virtual reality was first formulated decades ago and since then has evolved quite a bit, both regarding its definition as well as its practical application. Even today, the term virtual reality, including its exact definition and implications, is often used and understood differently between researchers, developers, and users. For instance, Bishop and Fuchs [27] defined VR as “real-time interactive graphics with 3D models, combined with a display technology that gives the user the immersion in the model world and direct manipulation”. Blach [28] describes that “VR-environments differ from conventional desktop in the sense that they embed the user in a computer-generated data environment” with the key properties of “3D-representation and perception”, “Spatial interaction in real-time”, and a “Sense of Presence and immersion”. Tsamitros et al. [383] names similar key elements of VR (condensation and reformulation of Sherman and Craig [336] original description):

1. Virtual world – “an imaginary space and a description of objects in a space and the rules and relationships governing these objects”
2. Immersion – “a state of being deeply engaged into an alternative reality/environment or point of view and/or the computer system’s technological capacity to deliver a vivid experience that removes the user from physical reality”

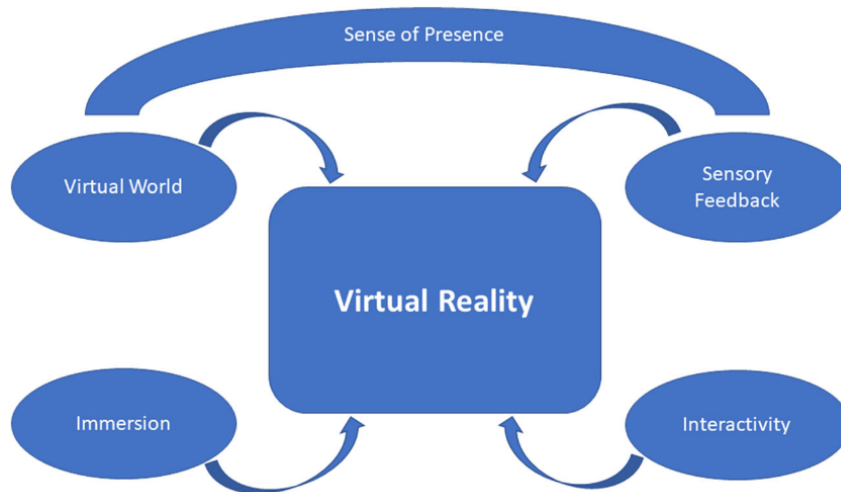


FIGURE 2.1: Key elements of VR according to Tsamitros et al. and Sherman and Craig [383, 336].

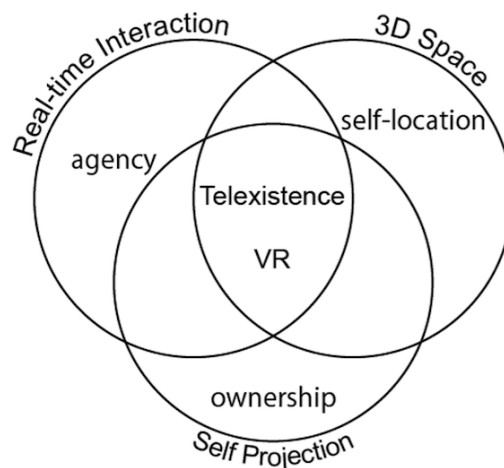


FIGURE 2.2: Key elements of VR according to Tachi [368].

3. Sensory feedback – “an essential ingredient of VR based on the physical positions of the participants (position tracking)”
4. Interactivity – “the ability of the participant to affect a computer-based world”

These elements are depicted in Fig. 2.1.

Similarly, Tachi [368]<sup>1</sup> identified the three key elements of VR that need to be fulfilled to be “three-dimensional spatiality”, “real-time interactivity”, and “self-projection”. Fig. 2.2 depicts the so-called “three-dimensional spatiality” that consists of these three elements.

VR can also be viewed as the latest manifestation of virtualization. In the real world, people can directly, and naturally perceive and interact with the environment. This environment can be modeled and simulated virtually. VR devices and special interaction metaphors then help to get multi-sensory perception of and multi-modal interaction within the virtual environment. Figure 2.3 depicts this concept.

<sup>1</sup><https://tachilab.org/en/about/virtualreality.html>



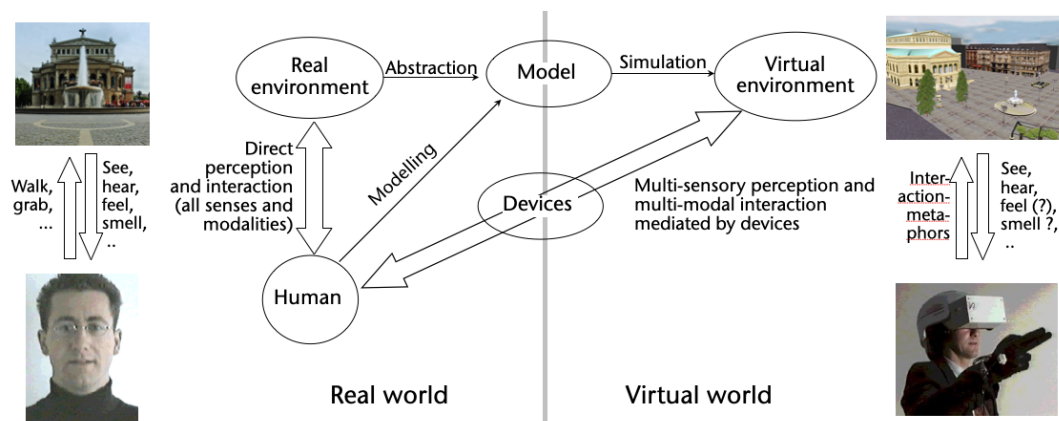


FIGURE 2.3: VR as the latest manifestation of virtualization. In the real world, humans directly perceive and interact with it. In VR, a virtual model of the real world can be simulated. Multi-modal/sensory interaction and perception are possible but require VR devices and special interaction metaphors.

As we see from the definitions, a VR system necessarily encompasses input and output devices to perceive the virtual environment and interact with it. Input devices could be simple ones such as joysticks, game controllers, and mouse and keyboard, or advanced ones such as motion trackers or bend-sensing gloves. Output devices, on the other hand, include speakers, haptic elements providing force feedback, and a wide range of visual devices, i.e., monitors, projector-based stereoscopic multi-wall environments (cave automatic virtual environment (CAVE)), and HMDs. [68] A HMD is a head-worn display device providing stereoscopic vision using two screens; each in front of one eye. Typically, they also include speakers and a positional tracking system. The most popular and prevalent VR setup consists of a combination of an HMD with two motion-tracked input controllers.

**Distinction between VR and AR/MR** Terms that get often mentioned in connection with virtual reality are augmented reality/mixed reality. VR aims at fully immersing the user in a completely virtual environment and tries to shut down the perception of the real world. In contrast, AR overlays computer-generated 3D content on the real world, enabling the user to perceive and interact with both the real world and the virtual content at the same time. [45] AR is usually experienced using AR glasses – think of a heads-up display –, but simpler and lower-cost devices such as smartphones and tablets can be used too.

### 2.1.2 Presence, Immersion, and Embodiment

Closely related to VR systems are concepts such as immersion, presence, and embodiment. However, these concepts are often confused with each other as there are many conflicting and ambiguous definitions for these terms and concepts.

**Immersion** Previously, we observed that immersion is often named a key element of VR systems. There is no consensus on its definition, though, as there are two major schools of thought. This makes immersion, in practice, a highly vaguely used term. One view is that immersion is defined by the technical quality of the system and

dependent on the VR system's properties, i.e., Field of View (FoV), screen resolution, or degree of motion tracking. If we follow this view, a VR system would be more immersive if it stimulates more senses, provides more interaction possibilities, and the provided stimuli are closer to the ones from reality. Thus, immersion would be an objective property of a VR system. [341, 344, 342] A proponent of this view is Slater, who formulated it this way [341]: "Let's reserve the term immersion to stand simply for what the technology delivers from an objective point of view. The more the system delivers displays (in all sensory modalities) and tracking that preserves fidelity in relation to their equivalent real-world sensory modalities, the more that is immersive."

In contrast, the other major perspective on immersion is that it is based on the user's psychological state, and therefore, a subjective property. Proponents of this view argue that the immersion is higher if the users are more involved, engaged, or absorbed by the virtual world. [3] For instance, Sanders and Cairns [316] defined immersion as "the sense of being "in a game" where a person's thoughts, attention, and goals are all focused in and around the game". Similarly, Thon [379] describes immersion as "a shift of attention from the real environment to certain parts of the game and the construction of a mental representation of the latter". Agrewal et al. [3] write that three major reasons for immersion (following the subjective school of thought) can be found in the literature: "the subjective sense of being surrounded or experiencing multisensory stimulation, absorption in the narrative or the depiction of the narrative, and absorption when facing strategic or tactical challenges". However, this view of immersion as a subjective property overlaps with many definitions of the term presence.

**Presence** Generally, the term presence describes the feeling of "being there" – being in another (virtual) place even though the actual physical location is different [152]. The term originated decades ago from the term **telepresence**, which, in turn, was used to describe teleoperation systems – machines that were operated or controlled by a human remotely from a distance [319]. Today, the term telepresence is mostly used, similarly to presence, to describe the perception of being and acting in a remote, mediated environment as opposed to the physical one [227]. Accordingly, technologies and systems that allow remote users to experience this, are called telepresence systems [293]. Presence is another key factor for VR systems. Similarly to immersion, for the term presence too, exist many definitions. For instance, Slater [341] considered it to be a subjective reaction to (the objective) immersion and defined it as the psychological perception of being in a virtual environment. Further, he argued that "presence consists of two orthogonal illusions that we refer to as Place Illusion (PI, the illusion of being in the place depicted by the VR) and Plausibility (Psi, the illusion that the virtual situations and events are really happening)" [346]. Patrick et al. [273] defined presence as "the extent to which a person's cognitive and perceptual systems are tricked into believing they are somewhere other than their physical location" and Schubert et al. [327] determined presence to be a product of the three components of "spatial presence", "Involvement", and "Realness". For a better distinction to related concepts and other types of presence such as social or co-presence, some authors use the narrowed-down term of spatial presence to describe the sense of being in a place [328].

The term **social presence** was originally introduced to describe a communication medium's ability to enable intimacy, immediacy, and interpersonal relationships

during a mediated conversation [339]. Intimacy refers to the feeling of connectedness between participants during an interaction, while immediacy is the psychological distance between them. Key factors for these properties would be nonverbal cues such as facial expressions and gestures. [263] In contrast to this medium-centric, technologically-determined concept, the term **co-presence** is often brought forward. It focuses more on the psychological interaction of the individuals [260] and refers to the “sense of being with another” [26]. To achieve a strong sense of co-presence, it is crucial to have a mutual awareness of the individuals, meaning, both, having a sense of feeling of other individuals as well as the other way round. [41]

**Embodiment/Virtual Body Ownership** Having an avatar, experiencing it as the own body, and, thus, feeling a sense of embodiment (or virtual body ownership) is an important aspect of VR and a significant contributor to the perception of feeling present in the virtual environment [345]. Gall et al. [120] define embodiment as the sensation that one’s self is located inside a virtual body, one controls this body, and that this body belongs to oneself. Previous research about the relationship between virtual avatars and the sense of embodiment found that not only the visual representation by and appearance of avatars are important for VR but also the degree to which the avatar is animated and follows the physical motions of the user has a significant impact [356]. According to Kilteni et al. [185], the sense of embodiment can be described as the sensation of “being inside, having, and controlling a body, especially in relation to virtual reality applications”. Accordingly, they state that the concept of embodiment can be separated into three components: the sense of self-location, the sense of agency, and the sense of body ownership. Self-location refers to the experience of being located at a body’s position/inside a body and at one specific location in space. Body ownership describes the feeling that a body is one’s own body and the body is the source of the experienced sensations. Agency, on the other hand, describes the feeling of controlling the body, its actions, and movements [185, 310].

## 2.2 Important Aspects of (Multi-User) VR

After we have defined in the previous section what exactly VR systems are and briefly talked about what key elements they consist of, in this section, we will go into more detail about the important aspects of (multi-user) VR. Specifically, we will look at the aspects from a technical side and present and discuss typical approaches and methods that are used to practically realize these aspects. Moreover, we will present concrete examples and talk about the current challenges of all these aspects. In the following, we will start by briefly talking about the crucial aspects of interaction and locomotion in VR systems (Sec. 2.2.1). Then we will discuss the topic of avatars, which we learned are important for representation and embodiment in (multi-user) VR (Sec. 2.2.2). We will also present sensing and tracking approaches (including extrinsic camera calibration) (Sec. 2.2.3), which are necessary components for a real-time scene or avatar representation in collaborative VR and telepresence applications. After that, we will give an overview of 3D reconstruction and rendering techniques (Sec. 2.2.4), discuss streaming and compression aspects with a special focus on image/video data and point clouds, as they are highly relevant for the visualization of remote scenes and avatars (Sec. 2.2.5), and lastly consider also the generation and visualization of the virtual VR environments (Sec. 2.2.6).

### 2.2.1 Interaction and Locomotion

Developing realistic and natural interaction techniques for VR applications is an important research topic, as natural interactions have been shown to be beneficial for usability, user performance, and embodiment [35]. The main types of interaction tasks are selection, manipulation, and locomotion/navigation [174]. The primary task of selection involves choosing a user interface (UI) element or other object in the virtual environment to further interact with and signaling this selection to the system. The task of manipulation concerns the interaction with a selected object according to its abilities and affordances (e.g., moving, rotating, resizing it, or pushing a virtual button, etc). Locomotion/navigation is about moving through the virtual environment.

Many selection techniques were developed but they can be divided into two main categories: virtual hand-based ones that closely mimic the natural process of directly/physically touching the object and pointing-based ones that enable the selection of objects at a distance [429]. Pointing can be done using the controllers, based on the head direction, or the gaze direction using eye tracking. The most popular pointing method is controller-based raycasting, however many variations and other techniques were proposed, e.g., crossing-based selection [386], or dynamically scaling potentially selectable objects [405]. The actual selection of an object is often implemented using a dwell time on target, clicking a button, or using a gesture such as pinching [422, 255].

Similarly, the task of object manipulation usually involves virtual hands mapped to the controllers and button presses. However, free-hand gesture manipulation is gaining attention lately; especially so in combination with gaze-based selection techniques [281, 216].

Navigation and locomotion is possibly the most crucial task in VR and, thus, got significant research attention. Recently, Di Luca et al. [85] identified over a hundred different VR navigation techniques. Reasons for the huge amount of competing locomotion techniques are that it is a challenging task to design a metaphor that provides a good user experience, and the need to make trade-offs between competing requirements such as accessibility and cybersickness [174]. An overview of the different locomotion techniques, categorized according to Martinzes et al. [237], and the degree to which they got scientifically explored can be seen in Fig. 2.4. Walking-based and steering-based techniques got the most attention, selection-based to some degree, too, while manipulation-based and automated techniques are only sparsely researched yet. Highly researched individual locomotion metaphors include the point-and-teleport technique [198] (a selection-based technique that is depicted in Fig. 2.5 (top left)) and the joystick-based steering [317]. Commonly used are also the walking-in-place [206] technique (see Fig. 2.5 (top right)), redirected walking [406] (see Fig. 2.5 (bottom left)), and head-directed steering. An example of a manipulation-based locomotion metaphor is the world-in-miniature technique [100] that is depicted in Fig. 2.5 (bottom right).

In collaborative VR environments, additional factors have to be considered to implement satisfactory navigation. For instance, when multiple people explore a virtual environment together as a group, an interesting challenge is to design a locomotion metaphor that allows the group to stay together and still provide valid and sensible locations for all the users. Various group navigation techniques were proposed for this, as described by Weissker et al. [403]. Another factor in collaborative VR which got not much attention yet is how to visualize discontinuous locomotion

### Categories of Locomotion Techniques

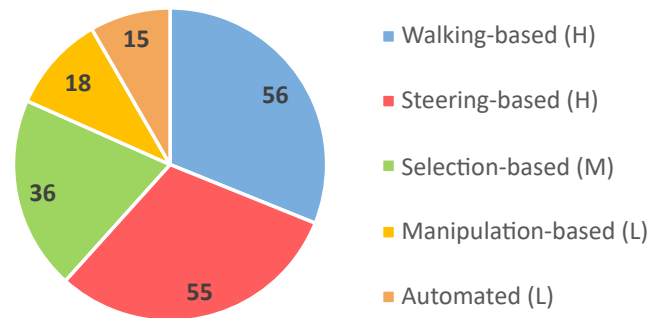


FIGURE 2.4: Categories of locomotion methods and the degree of their scientific exploration, vgl. [237]. Walking- and steering-based locomotion are highly (H) researched, research on selection-based metaphors is medium (M) and on the rest low (L).

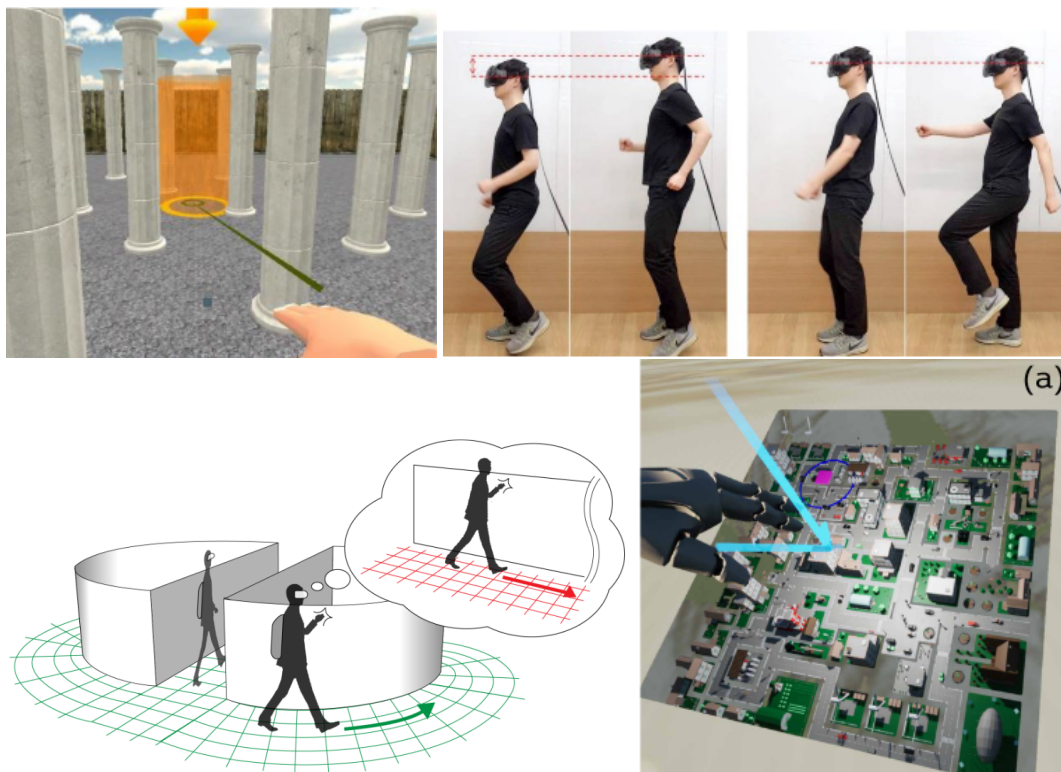


FIGURE 2.5: Various locomotion techniques. Top left: simple point and teleport [36]. Top right: walking in place; the tracked rhythmic movements from the walking motion drive the virtual locomotion [206]. Bottom left: redirected walking; virtual long distance locomotion is achieved by redirecting curved real-world paths [240]. Bottom right: world in miniature, which is similar to a 3D mini-map [100].

techniques, such as the highly popular point and teleport metaphor, that provide no motion cues to observers.

### 2.2.2 Avatars

The visual representation of the users in the virtual environment through avatars is a highly important aspect of (collaborative) VR and accordingly a big research topic. Studies showed that visual realism and appearance as well as behavioral realism and extent/precision of motion tracking are important factors affecting the sense of embodiment and presence. For instance, Thalmann and Thalmann state that to create believable virtual humans, one needs not only realistic appearance modeling but also realistic, smooth, and flexible motion modeling [375]. Young et al. [428] found that avatars that do represent the full body, and not only parts of it, are beneficial for collaboration in VR. Similarly, Wang et al. [398] found full-body avatars to be advantageous for remote instruction scenarios, and Latoschik et al. [203] showed that realistic looking avatars are crucial to increase the sense of embodiment in VR. Moreover, Waltemate et al. [392] reported that personalized (full-body) avatars significantly increase the sense of embodiment and presence compared to a generic one that otherwise has the same level of realism. Recently, Wu et al. [416] showed also that highly expressive avatars, which include tracking and visualization of full-body movement, hand gestures, and mimics, increase the sense of social presence, attraction, and task performance.

Over time many different types of avatars, and, accordingly, various more or less sophisticated tracking and reconstruction techniques were developed. This vast design space of 3D avatars can be categorized by multiple important factors/criteria:

- Extent of representation: The degree to which the body is represented by the avatar. On the one end is for example the popular so-called “floating hands” model that consists only of a head and two floating hands, see the left image of Fig. 2.6. On the other end of the spectrum are full-body avatars.
- Extent of visual realism and personalization: How realistic the avatar looks visually and how closely it resembles the actual user.
- Extent of visual interactivity: This can range from (realistic but) appearance-wise static pre-constructed models to real-time reconstructed/updated avatars where the textures, mimics, or wrinkles in the clothes change dynamically, see the middle and right images of Fig. 2.6, respectively.
- Extent of behavioral realism/tracking: How realistic and extensive the movements of the user are tracked and realized in the virtual world, e.g., three-point tracking of only head and hands or full-body motion tracking of all joints.

### 2.2.3 Sensing and Tracking Methods

In order to visualize remote scenes and especially people in real-time, they have to be captured and tracked beforehand by appropriate sensors and methods. Over the years, many approaches emerged with different advantages, limitations, and goals. For instance, a popular approach is to employ marker-based systems such as OptiTrack<sup>2</sup>. For example, Kasahara et al. [178] employed this approach to investigate the effects of body deformation on the sense of embodiment. These systems rely on

---

<sup>2</sup><https://optitrack.com/>



FIGURE 2.6: Various avatar types. Left: A “floating hands” avatar. Middle: An animated full-body avatar. Right: A live-captured and reconstructed full-body avatar.

reflective or LED markers that get attached to objects, clothes, or a worn bodysuit, see Fig. 2.7 (left). The markers then get tracked by an array of infrared (IR) cameras and can be used to reconstruct a skeleton from the marker positions [49]. The advantage of this approach is that it is very precise, however, marker-based tracking can be uncomfortable for the users and is comparatively expensive [208]. Occlusion of markers is an issue, too, but can be managed by careful placement and redundancy. Another simpler approach is to use the HMD and accompanying motion controllers to track and reconstruct the user’s motion. In this case, there are only 3 sources for the tracking (therefore it gets often referred to as 3-point tracking) which makes the comprehensive reconstruction of the user’s motion difficult. Usually, this is attempted by using inverse kinematics and animation blending, as done by Jiang et al. [162], or Parger et al. [269]. Additional trackers can be used to also track for example the feet [48]. To avoid the limitation of having to attach markers, marker-less tracking is also a big research topic. Marker-less approaches use cameras to capture the scene and directly 3D-reconstruct it or employ computer vision and body pose estimation algorithms to track the users explicitly, see Fig. 2.7 (right). Some works rely only on color cameras while most use stereo or RGB-D cameras such as the (Azure) Kinect by Microsoft<sup>3</sup> that also provide depth estimates [49]. In camera-based tracking, inconsistent tracking [239], occlusion [114], and high latency [34] are challenging issues. To avoid occlusion and enhance the robustness, again, multiple cameras can be set up that capture the scene from multiple angles [352, 208]. To better track hand gestures, camera-based tracking with devices such as the Kinect is sometimes combined with additional optical tracking sensors such as the Leap Motion controller<sup>4</sup>, which technically is a pair of IR cameras. Attached to the VR headset, these are better suited to track fine details such as the hand and finger poses. For instance, Lin et al. [221] combined a Kinect camera with an HMD and a Leap Motion controller for detailed pose tracking in robot motion planning. Another issue when employing camera-based tracking is when the tracked person wears an HMD (e.g., in telepresence applications), as it blocks the view of the face and prevents a live capture of the mimics. To tackle this issue, many recent HMDs include additional eye/face tracking sensors.

<sup>3</sup><https://azure.microsoft.com/de-de/products/kinect-dk>

<sup>4</sup><https://www.ultraleap.com/product/leap-motion-controller/>

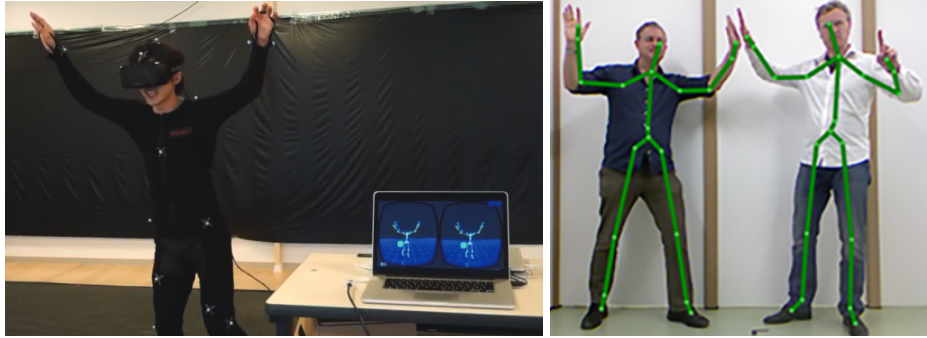


FIGURE 2.7: Example of marker-based motion tracking [178] (left) and pose estimation using a RGB-Sensor [126] (right).

When multiple sensors, i.e., RGB-D cameras, are employed to enhance the robustness and scene coverage, they have to be accurately registered to each other (extrinsic calibration) to precisely merge the data. RGB-D camera calibration has been widely researched. The usual approach is to use an easily-detectable registration target such as a checkerboard for calibration [94, 435]. However, many other methods were proposed: e.g, using 3D targets that are also visible in the depth images [248, 80], using no explicit targets at all but using local and/or global algorithms to directly match point cloud reconstructions [349, 211], or to employ additional external tracking systems [19].

### RGB-D/Depth Enhancement

As we briefly discussed previously in Section 2.2.3, RGB-D cameras get popularly employed for live scene capture, but the raw data suffers from various issues such as noise and artifacts that are caused by inconsistent tracking and occlusions [381]. Thus, enhancing RGB-D data, and specifically the depth images, is another important research area. One crucial requirement for any developed enhancement algorithm is that it can be applied in real-time, meaning with at least the 30 Hz the RGB-D sensors usually capture with. We can divide the area of RGB-D enhancement into two categories, based on the concrete issue that is targeted. The first category is denoising of the depth data, and the second one is depth image completion. Image and signal denoising is a classic problem. Thus, it has been studied for a long time. Popular methods, for instance, are Kalman filters [396, 139] and bilateral filters [168, 268]. Moreover, as the input usually is a continuous stream of captured images, denoising methods have not only to consider the spatial domain but also the temporal one to achieve the best results [40, 419]. Recently, deep learning-based image and video denoising methods achieved impressive results [371, 338]. However, depth images, and their noise characteristics, are inherently different from regular RGB images and videos, making customized denoising algorithms [251, 61] and network models necessary [358, 89]. The other main task in depth enhancement is depth image completion/inpainting or hole-filling. Depth images inherently suffer from holes – areas without valid data –, which are caused by various issues with the capturing methods, such as multi-path inference, see Fig. 2.8. Similarly to the denoising task, the field of image inpainting and restoration is extensively researched. Traditionally, image inpainting was often done using pixel- or patch-based methods, see for instance the work by Ruzic et al. [314], however, also in this area are deep-learning methods on the rise [291]. Especially U-Net-based convolutional neural networks (CNN)s with non-standard convolution layers, see for example the work



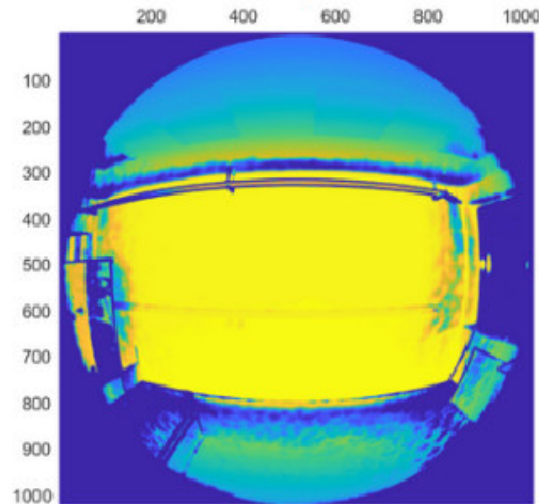


FIGURE 2.8: Depth image from the Azure Kinect RGB-D camera. Note the blackish/bluish areas where no valid data has been captured [381].

by Liu et al. [223] and transformer networks such as the one by Deng et al. [81] seem to be promising for image inpainting. However, works specifically targeting depth images are rare. And the ones that do consider depth images, mostly have other limitations, e.g., that they rely on color image guidance [115, 232], or handle only smaller holes [163].

## 2.2.4 Reconstruction and Rendering Methods

3D reconstructing and rendering of (pre-processed and enhanced) sensor or tracking data is a crucial element to have real-time avatars and surrounding scenes in telepresence applications. Especially challenging is the real-time reconstruction of dynamic scenes and moving/deforming objects. As discussed in Section 2.2.3, there are different approaches to sensing and tracking that practically dictate which techniques for reconstruction and rendering make sense. We can distinguish between two main approaches: usually if marker-based tracking or explicit pose- and skeleton tracking/reconstruction are employed, they are used to drive/animate pre-constructed (mesh) objects and avatars. These may be constructed using photogrammetry, laser scans, and highly complex, sophisticated reconstruction algorithms, see for example the right image in Fig. 2.9. Works that follow this approach are for example the ones by Wenninger et al. [408] and Bartl et al. [18]. However, with these methods, the objects and avatars tend to not reflect real-time changes in texture and fine details that are not reflected in the tracking data. Also, the setup to initially scan the person of which the avatar should be created is traditionally very complex and expensive, although some more recent techniques try to mitigate this limitation by limiting themselves to requiring fewer and cheaper cameras or just a short video of the person [161]. Bartl et al. [18] did a descriptive comparison of high-cost and low-cost offline 3D reconstruction. On the other hand, if no explicit tracking is done but raw RGB-D data is used, then rendering simple point clouds, e.g., using splatting [33, 299], is a popular solution, see for example the middle image in Fig. 2.9. Yu et al. [431] compared this approach with animated mesh-based avatars and found this to be a feasible solution. Another option would be fast point cloud-based real-time 3D mesh reconstruction, in the most simple case this could be done



FIGURE 2.9: Various scene/avatar reconstruction methods. Left: real-time volumetric RGB-D-based mesh reconstruction [91]. Middle: real-time point cloud rendering [431]. Right: high-cost and low-cost offline 3D mesh reconstruction [18].

by connecting/triangulation of neighboring points in an ordered/organized point cloud [164, 147]. Naturally, then the objects and avatars reflect the exact appearance at all times but are less detailed as they have to be generated in real-time [408]. When live reconstructing avatars using point clouds etc, we have the issue of the face being obstructed by the HMD, as mentioned in Section 2.2.3. This can be solved, however, by making use of the eye/face trackers often included in the HMDs and approximating the eye movements and mimics using, for instance, animations and deformable (pre-personalized) meshes of the face/head [290]. Another group of 3D reconstructing and rendering methods is volumetric reconstruction, sometimes called volumetric performance capture, which creates a volumetric representation (that often includes temporal fusion throughout consecutive frames) before eventually rendering the final mesh (left image in Fig. 2.9), as done by Dou et al. [91], Guo et al. [137], Su et al. [362], and Cho et al. [65]. The latter explicitly compared this approach with pre-scanned mesh avatars. and rendering using implicit surfaces and raycasting. Generally, the reconstruction and rendering results are highly dependent on the employed technique, the available time and resources, as well as the sensor accuracy.

Moreover, depending on the application's domain, there may be additional data that would profit from being visualized and inspected in 3D. In the medical field, for instance, CT and magnetic resonance imaging (MRI) scans are commonly used for diagnosis and surgery planning. These can be visualized using projection techniques but volume rendering methods proved to be helpful, too, as demonstrated by, for example, Wang et al. [395]. Volume rendering methods can be divided into indirect volume rendering, also called surface shaded display, and direct volume rendering, which became the more popular method in recent years [130]. The former provides realistically looking 3D views by creating a polygonal surface representation of the volume while the latter directly visualizes the whole volume of data by, for instance, accumulating opacity and color using raycasting. The work by Jung et al. [170] is a good example of how direct volume rendering can be employed to visualize PET-CT scans. Recently, even more sophisticated methods were proposed that achieve very high-quality visuals, however, are also very computationally demanding [22]. Integrating these volume rendering techniques in collaborative VR applications is still a challenge, although a few works, such as the one by Maloca et al. [233], were presented that provide volume rendering solutions in VR.

### 2.2.5 Streaming and Compression of Sensor Data

An important aspect to consider when developing multi-user VR applications, especially when using RGB-D sensors for live capture, is the necessary amount of data that has to be streamed in real-time to or between the users. Image or video streams as well as point clouds require a huge amount of bandwidth if no compression is applied [364]. Therefore, fast and efficient compression of these data formats is another important research branch [226]. Naturally, the goal is to achieve compression ratios as high as possible to reduce the amount of data that has to be streamed. On the other hand, stronger, more complex compression algorithms require more time and computational power for compression and decompression, which raises the users' hardware requirements, and more importantly, increases the latency. The latter is a crucial factor in VR applications, though. Therefore, compression algorithms need to be highly efficient, find a good balance between compression ratio and required time, and ideally be flexible in this regard. Moreover, depending on the application domain, the compression has to be lossless (or near lossless), e.g., when employed in remote surgery assistance scenarios. One approach is to compress and stream the computed meshes or point clouds. This is usually done by using hierarchical, spatial data structures such as octrees [93, 187], plane fitting, or combinations of both [92]. Some works do also consider the temporal domain by comparing the changes between frames [108, 176]. Most of these geometric compression methods are lossy. Pereira et al. [277] recently gave a good overview of these techniques. Another approach is to directly compress the captured color and depth images. Image and video compression is a long-standing and well-established research topic with many proposed algorithms and methods, lossy and lossless ones. Thus, it is natural to consider applying these tried and tested methods/codecs, as done by Tu et al. [385]. Common ones are, for instance, JPEG, PNG, H.264, H.265, AV1, and many more. However, one crucial issue with this idea is that the depth images are inherently different from color images (e.g., the number of channels, the bit depth, the sparsity of the data, or the abrupt changes at object edges), therefore, adapted codecs as proposed by Pece et al. [275] or novel solutions such as the RVL algorithm by Wilson are necessary for the best performance [412]. Other examples of dedicated depth-image compression methods, that usually incorporate classic (image/video) compression methods such as clustering/segmentation, arithmetic coding, or delta coding, are the works by Sun et al. [364], or MPEG's V-PCC [158].

### 2.2.6 Virtual Environments

When discussing important aspects of (multi-user) VR systems, the virtual environment itself cannot be overlooked. The virtual environment of a VR scene consists usually of mesh-based 3D objects that get arranged as needed to build the scene or level that is required for the specific application. As described earlier, in some use cases, the virtual scene can be rendered partly or even completely using live capture and 3D reconstruction. One example could be a VR teleconferencing application in which not only the persons themselves but also the office, desk, etc. get live captured using RGB-D cameras [361]. Figure 2.10 depicts an example of an environment that gets live reconstructed using multiple RGB-D sensors and a point cloud representation. In other cases, a real environment gets (pre-)scanned, e.g., using cameras and photogrammetry, or laser scanners, and virtually reproduced [425]. These scans then can be rendered using point clouds or mesh reconstruction [186, 324]. However,



FIGURE 2.10: Example of a small-scale environment, which gets live-captured by multiple RGB-D sensors and rendered using point clouds [361].

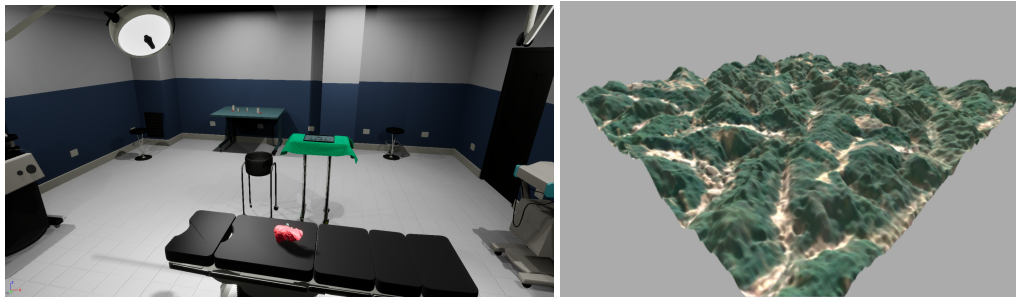


FIGURE 2.11: A small-scale, manually crated mesh-based VR environment on the left side and a procedurally generated large terrain on the right side [351].

most often the virtual environments consist of static, manually modeled mesh objects, see for example the left image of Fig. 2.11 that shows a virtual operating room. The latter gets problematic when the application requires a big, complex environment such as a huge, detailed terrain or landscape. This is sometimes, for instance, the case with virtual testbeds and simulation environments [201, 378, 215]. Generating these large, detailed landscapes manually is a laborious task [6]. Therefore, much effort is put into procedural generation methods that can automatically create these landscapes, see for example the right image of Fig. 2.11 that shows a large procedurally generated terrain. Although many methods were developed over the years, it is still a challenging task to develop algorithms that produce realistically-looking, high-quality terrains while still providing flexibility and intuitive control to the designer.

The many proposed approaches of procedural terrain generation can be broadly classified into synthetic methods, physically-based methods, and example-based methods [144]. Synthetic methods such as noise are very popular as they are quick and easy to use, however, it is hard to get realistic results [149]. On the other hand, physically-based methods try to replicate the physical, geologic, and other natural processes to get highly realistic results. Common are for instance various erosion simulations [157, 13]. The drawback of these methods is that they are slow and not as flexible. Example-based methods are also able to create realistic terrains using real-world example images, i.e., DEMs, [442, 135]. Naturally, they are limited to having specific examples for the desired terrain.

## 2.3 Example Application Domains

In this section, we will give an overview of the diverse application domains of VR and present various example applications. This will help to get a better understanding of the many crucial VR aspects that we presented in Section 2.2, how they come together in fully-fledged applications, and how they are used in practice.

The ability to virtually depict arbitrary objects and environments that can be viewed and explored from home makes VR a highly useful tool in the area of cultural heritage. The utilized technologies to digitally preserve and faithfully reproduce historical buildings or environments in VR range from classical 3D modeling and photos, to photogrammetry and laser scanning [53, 225]. With these technologies, the natural interaction provided by VR, and the ability for additional data augmentation, the development of immersive VR experiences that allow one to explore virtual environments gets increasingly popular. One example is the VR application *Nefertari: Journey to Eternity* [104] which allows individual users to explore a virtual reproduction of the tomb of Nefertari using VR. Users can freely explore the tomb using the teleportation metaphor and interact with interactive elements using virtual controllers, see Fig. 2.12. The virtual environment was re-created in high detail using photogrammetry. Data post-processing included hole-filling and denoising. Similarly, the Palace of Versailles was virtually reproduced, also using photogrammetry, and is explorable in the VR application *VersaillesVR the Palace is yours* [132]. Here, users can move using teleportation and the world-in-miniature metaphors and select and interact with objects using virtual controllers and virtual pointers. Again, only individual exploration is supported. Another similar historical VR recreation based on photogrammetry is *IL DIVINO: Michelangelo's Sistine Ceiling in VR* [103]. In contrast, the application *IL Gigante: Michelangelo's David in VR* [102] did rely on extensive laser scanning to digitally reproduce a high-fidelity model of Michelangelo's David and Marzouk et al. [238] did also use laser scans to virtually and faithfully reproduce an Egyptian palace. Both of them did use the point cloud representation resulting from the scans to do registration of the individual scans and other processing tasks such as noise removal but eventually created a 3D mesh out of the point cloud. Apart from the field of cultural heritage, VR-based tools are also an upcoming solution for medical imaging visualization due to their advanced and immersive 3D visualization and interaction abilities [283]. Two examples are the multi-user VR application by Prodromou et al. [288] that visualizes MRI data as interactive 3D volume-rendered models in VR and the work by Scholl et al. [321], who presented a VR system that provides 3D representations of CT or MRI data using direct volume rendering. Interaction with the volume data is possible using the VR controllers and direct touch, gesticulation, and a 3D UI.

Multi-user virtual reality systems are also getting increasingly popular to build immersive telepresence tools that can be used for tasks such as telementoring and teleconsulting [98]. In the medical area, for instance, remote surgery assistance and consultation using collaborative VR/AR tools is an upcoming topic, as briefly discussed in Section 1.1. The natural interactions provided by such systems are a big advantage but the main one is probably the ability to 3D reconstruct and visualize live-captured remote scenes and people when combined with suitable capturing and tracking systems. For instance, Gasques et al. [124] proposed a collaborative mixed-reality system for immersive surgical telementoring. In it, remote experts use VR to guide local novices, which in turn use AR glasses. Using RGB-D cameras



FIGURE 2.12: VR application that accurately recreates Nefertari's tomb using photogrammetry and enables users to freely and immersively explore it [104].

and the OptiTrack marker-based tracking system, the patient and other relevant objects get registered with each other and real-time reconstructed in the virtual environment using a point cloud representation. Also using OptiTrack and additional inertial measurement unit (IMU)-equipped gloves, and a tracked pen, the expert, his gestures, and annotations get tracked and transmitted to the novice. The novice and expert get visualized to each other as tracked, un-personalized avatars. The novice's avatar consists just of a floating head and torso, while the expert's avatar additionally has floating hands. The expert can interact and annotate objects using the tracked pen that can be used either using direct touch or as a laser pointer. Anton et al. [9] developed a similar system for real-time remote medical consultation that is based on AR and VR. On the patient side, AR is used while the expert side employs VR. With the help of an RGB-D camera, the patient's body gets live captured, which then gets transmitted for reconstruction (3D mesh) and visualization to the remote expert. The expert, in turn, can make annotations using a stylus and a stereoscopic display that can be observed on the patient's side. Data communication is implemented using the WebRTC protocol and 8-bit YUV-based compression for the depth data. Another example of a mixed-reality-based telepresence system is the one proposed by Maas et al. [228]. It also features an RGB camera for live capture and 3D point cloud reconstruction as well as an AR headset on the patient side and a VR system on the remote expert side. The proposed MR telepresence system by Strak et al. [359] follows the same principle. 3 RGB-D cameras are used to live-capture and 3D reconstruct the patient and his environment for the expert in VR. The left image of Figure 2.13 depicts the virtual scene including the reconstructed environment and the remote expert's avatar. The top right image shows the remote expert's view of the live-reconstructed patient, and the bottom right image shows the remote expert's avatar as seen by the local expert. Depth compression of the data is done losslessly using H.264. On the expert side, the data gets reconstructed into point clouds and eventually into a 3D mesh. The expert can make annotations using the VR controllers, which are visualized on the patient side. The expert is represented by a 5-point tracked full-body avatar that also includes mimics captured by eye- and face-trackers. More works that presented MR- or VR-based systems for remote collaboration are the ones by Piumsomboon et al. [285] and Wu et al. [416]. Both works did focus on the avatar representation. The former combines AR and



FIGURE 2.13: A MR telepresence application. a: third-person view of the real-time 3D-reconstructed environment including the remote expert's avatar (right). b: first-person view of the remote expert on the virtual patient. c: remote expert's avatar as seen by the local participant. [359].

VR and visualizes participants as tracked full-body avatars (AR users get tracked by the AR glasses and VR users by the HMD and the controllers). The latter only used VR but features a sophisticated tracking system to create highly expressive avatars. Specifically, multiple RGB-D cameras, the HMD, and multiple attached Leap Motion tracking devices are employed for body and hand tracking and to drive and animate a full-body avatar.

One of the biggest application domains of (multi-user) VR is education and training. Using VR, virtually anything can be trained in a safe and cost-efficient manner. This can be done individually, collaboratively, or in a student-teacher scenario. In the case of the latter options, the VR system would also bring in its telepresence capabilities. Some examples of the vast amount of VR training and education systems that were developed and proposed are the Immersive Anatomy Atlas by Gloy et al. [131], or the dental training simulator by Kaluschke et al. [172]. The former consists of a hand-modeled, mesh-based virtual environment and a detailed anatomical 3D model of a human. The anatomy model consists of individual organs and structures, which are again, rendered as 3D meshes. Each organ or structure is interactive and can be freely grabbed and moved around. The user is represented by a virtual HMD and a pair of virtual hands, which are tracked by the HMD and the controllers. Movement is implemented through room-scale and teleportation and selection and manipulation by directly touching an object and pressing a button on the controller. Additionally, the user has a range of virtual surgical instruments at his disposal, that can be used to cut or annotate the model. A similar VR anatomy education application was proposed by Schott et al. [322]. It specializes in liver education and, in contrast, provides multi-user and AR support. Another example from the medical area is the collaborative VR laparoscopic liver surgery training environment by Chheang et al. [63]. It enables multiple users to meet in a shared virtual environment and manipulate the same volumetric patient organ model (which is converted and rendered as a mesh in real-time), see Fig. 2.14 (left). For object interaction and manipulation, VR controllers are used as well as additional laparoscopic surgical joysticks, which can be seen in Fig. 2.14 (middle). Users can explore the 3D organ and virtually plan and perform surgical interventions in an exploration mode, or train and virtually laparoscopic procedures using the surgical joysticks in a virtual operating room in the surgery mode, see Fig. 2.14 (right). Other examples of VR

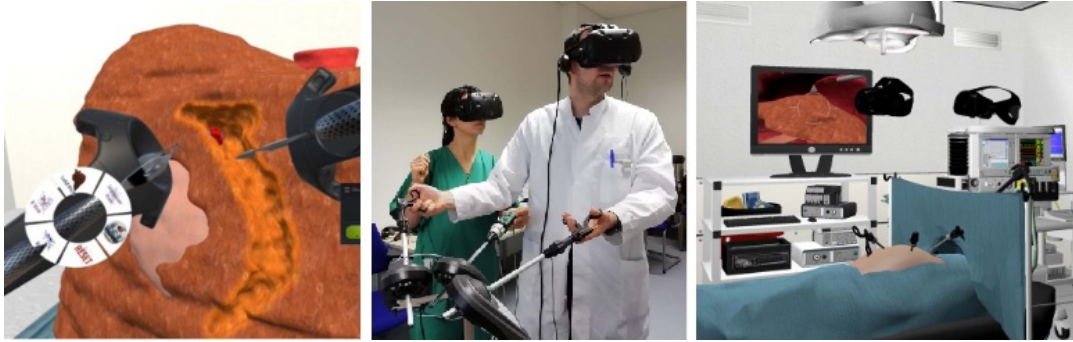


FIGURE 2.14: Collaborative VR training environment for laparoscopic liver surgery. Left: training and planning surgical procedures on a virtual volumetric model. Middle: users interacting with surgical joysticks and VR headsets. Right: virtual, collaborative training environment. [63].

training environments from the medial area can be found in the works by Roy et al. [312], Pfandler et al. [280], McGrath et al. [242], and Anbro et al. [7]. VR systems for education and training are also used in other domains, for instance, for pilot training [298, 262, 284], construction safety training [315], or multi-user industrial training and education [8]. The latter provides various virtual environments, e.g., a production line of engine assembly, in order to teach high-level processes in the automotive area. The environments also consist of hand-modeled meshes. The users are represented by animated (un-personalized) full-body avatars.

Somewhat related to training in a virtual environment is the use of VR as a simulation environment or virtual testbed. However, here the focus does not lie in educating and training the user on specific procedures but in testing systems and algorithms. Again, the virtual environment provided by the VR system has huge advantages regarding feasibility, costs, observability, and safety. For instance, Yao et al. [426] proposed a VR-based virtual testbed for autonomous-driving vehicles in which the autonomous vehicles and the corresponding algorithms can be trained and tested safely on structured and unstructured roads and the results can be directly observed in VR. The virtual environment, including the road networks, is generated automatically, as handcrafting kilometers of roads would be too laborious. Moreover, Teuber et al./Weller et al. [407, 374] developed a virtual testbed for planning and simulation of planetary swarm exploration missions. Using VR (HMD or powerwall + OptiTrack motion tracking), multiple users can together investigate and observe simulated sensor output and interact with the world and the autonomous agents. Interaction is done using direct touch, ray selection, a 3D graphical user interface (GUI), and teleportation. The users get represented by animated, un-personalized full-body avatars and the terrain is based on satellite data that was augmented semi-automatically by additional surface details. In contrast, Vitacion and Liu [388] developed a VR system that fully procedurally creates three-dimensional geographical/spherical surfaces for space mission simulation. The terrain is generated using noise and subdivision techniques and can be observed directly in a VR environment. Looking at the industrial and manufacturing area, VR simulation environments also get successfully employed for tasks such as industrial workstation robot design. As an example, Havard et al. [141] developed a multi-user VR-based simulation environment for the design and assessment of industrial workstations, and Togias et al. [382] presented a VR environment for industrial robot



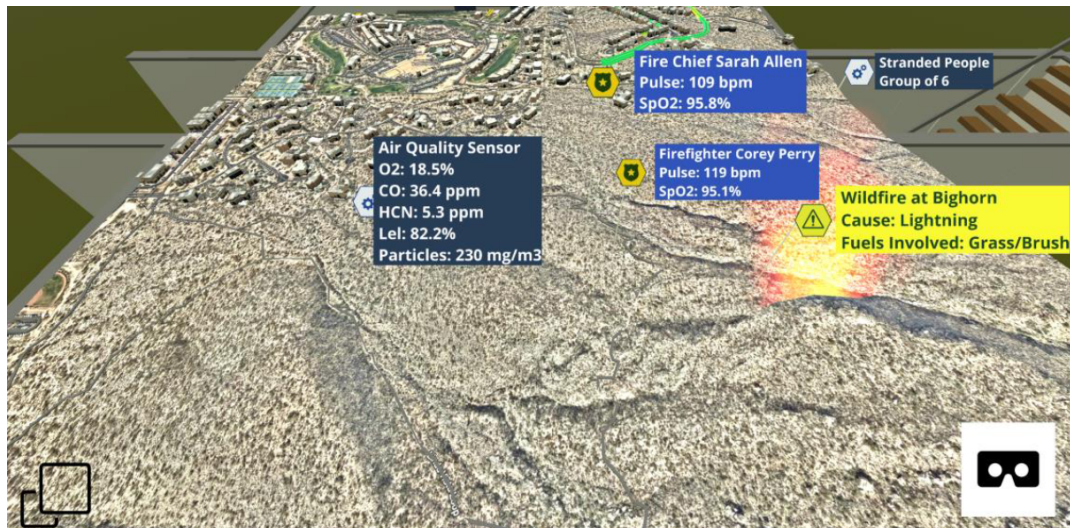


FIGURE 2.15: Collaborative VR simulation environment with automatically generated terrain and information overlays [332].

control and path design. There, users can freely move through the virtual shop floor using joystick movement, interaction is provided by virtual buttons on a GUI and natural interactions using the VR controllers. Each programmed action and the resulting paths are visualized in the virtual environment. Xie et al. [421] developed a virtual testbed for realistic human-robot interaction. It provides pipelines to create or import 3D scenes or models from a variety of sources and in various formats, e.g., meshes, RGB-D data, etc. Humans get visualized with full-body mesh-based avatars that get tracked using an RGB-D camera and pose estimation, the HMD, and a dance pad for navigation. Input and manipulation are realized using a VR controller, hand-tracking devices such as Leap Motion, and tracking gloves. Virtual environments are also popularly used as simulation environments for ecological and environmental simulations. For example, Sermet et al. [332] proposed a VR framework for collaborative environmental simulations, as can be seen in Fig. 2.15. It provides dynamic visualizations such as water and fire simulation, as well as information layers for disaster damages, traffic, and weather. The 3D environment is generated using geo-streaming services such as Mapbox and dynamically overlaid with additional information and effects. Cirulis et al. [69] proposed a VR-based simulation environment for bog ecosystem simulations. The terrain is automatically created using real-world heightmaps, while the trees are placed or destroyed dynamically according to the simulation or direct interaction by the user. Similarly, Weller et al. [404] presented a VR-based simulation environment of a coral reef, in which the 3D corals are placed and grown procedurally according to a simulation.



## Chapter 3

# Algorithms and Architectures for Telepresence in Multi-User VR

As discussed in Chapter 1.1, telepresence plays a major role in collaborative and multi-user VR applications. For instance, immersive and high-quality VR-based telepresence systems could be of great benefit in the medical field, as they allow distant experts to interact with each other and to assist local doctors as if they were physically present. Naturally, participants in such systems have to be represented using virtual avatars that faithfully depict the physical users. Streaming, reconstructing, and rendering the live-captured scene and avatars in real-time and high quality is a challenge, though. The data captured by RGB-D cameras, specifically the depth images, consume a lot of space and quickly saturate the available bandwidth of common connections. Moreover, the recorded depth data suffers from noise and artifacts, leading to holes in the images, and in turn, the eventual 3D reconstructions. Hence, in this chapter, we focus on these important aspects by presenting a series of new algorithms and architectures. First, in Section 3.1, we present an all-encompassing pipeline for point cloud streaming and rendering for multi-user VR that provides participants with real-time 3D reconstructions of the physical scenes and avatars of the remote users. As an example use case for our streaming pipeline, we chose to focus on remote surgery assistance, although the work can easily be applied in other areas too. Then, in Section 3.2, we present a lossless depth image compression algorithm that is able to reduce the required bandwidth significantly over the common RVL algorithm but still achieves real-time speed. Lastly, Section 3.3, is dedicated to further enhancing the depth images, and thus the resulting avatars, by applying adapted deep inpainting models from the color-image domain that quickly reconstruct the missing areas.

### 3.1 Development and Evaluation of a Point Cloud Streaming and Rendering Pipeline

Despite recent advances in VR technology, and more telepresence systems making use of it – we presented a broad overview in Section 2.3 –, most of the current telepresence solutions in use (if any), are just video-based and don't provide the feeling of presence or spatial awareness, which are highly important for tasks such as remote consultation, -supervision, and -teaching. Reasons still holding back VR telepresence systems are high demands regarding bandwidth and computational power, subpar visualization quality, and complicated setups. In fact, most of the issues discussed in Section 1.1 apply to VR telepresence systems.

Therefore, in this section, we chose the example application domain of remote surgery assistance and propose an easy-to-set-up VR-based telepresence system that

enables remote experts to meet in a multi-user virtual operating room. There, they are able to view live-streamed and 3D-visualized operations, interact with each other, and collaboratively explore medical data. Our system is based on Azure Kinect RGB-D cameras, a point cloud streaming pipeline, and fast point cloud rendering methods integrated into a state-of-the-art 3D game engine. Remote experts are visualized via personalized real-time 3D point cloud avatars. For this, we have developed a high-speed/low-latency multi-camera point cloud streaming pipeline including efficient filtering and compression. Furthermore, we have developed splatting-based and mesh-based point cloud rendering solutions and integrated them into the Unreal Engine 4. Finally, we conducted two user studies with doctors and medical students to evaluate our proposed system, compare the rendering solutions, and highlight our system's capabilities.

The work presented in this section is based on our published paper PC2 in Appendix A.

### 3.1.1 Introduction

Telemedicine plays a major role in medicine and health care and, although it is hardly a novel concept, it has received increased attention lately. As discussed in Section 1.1, the ability to provide assistance from a distance, and collaborate without the need for being physically present, exhibits a huge potential to provide patients with better care, increase efficiency, and save costs [111, 193]. There is a wide range of applications for telemedicine and the more advanced telepresence systems: from telementoring and training, remote consultation and collaboration, to remote diagnosis, surgery, and rehabilitation [305, 9]. We previously presented the exemplary application of remote surgery assistance in Section 1.1, but we want to give a brief recap here: The scenario is emergency situations with traumatic injuries where fast interventions are critical. Regularly, the dilemma is to either spend valuable time transporting the patient to specialized health care facilities, or to go to the nearest hospital although the local surgeons, especially in rural areas, might be less experienced [95]. These local surgeons could benefit from consultation or even mentoring from remote experts via telepresence systems that could preemptively be integrated into the surgery rooms [202]. To give another example, VR telepresence systems could also be employed in order to reduce health risks by limiting physical contact with possibly contagious patients and medical staff to a minimum; novice surgeons could consider attending surgeries via telepresence instead of being physically present in the operating room [79]. Other applications could be patient visits/ward rounds in intensive care units, or tumor conferences where normally many experts from different medical areas come together to discuss the situation and the further procedure [311].

Telemedicine in the past, and to a significant degree today too, relied mostly on classical video conferencing systems and other video-based solutions [11, 159]. These systems are inherently limited by the fixed point of view, lack of depth perception, and 2-dimensional screens, preventing a distinct feeling of (tele-)presence [331, 16]. As briefly mentioned in the beginning, continuous technological advances and the emergence of improved, affordable VR/AR devices lead researchers to focus on 3D VR/AR-based telepresence solutions. These systems are intended to deliver a more immersive experience compared to older video-based solutions, enable more natural interactions, and provide a better spatial understanding of the objects and their surroundings [294]. More details about a number of example applications can be found in Section 2.3 and in the following related work section. Many studies

showed that in such systems the users' representation through personalized high-quality avatars is fundamental [122, 65, 431]. To create virtual 3D representations of the scene, these systems usually employ RGB-D cameras or other depth sensors, whose data then has to be streamed to the remote location to be viewed in VR. However, VR-based telepresence systems still face several challenges: high bandwidth requirements for transmission of the data, inadequate real-time 3D reconstruction and rendering quality, or hard-to-set-up systems. We discuss these aspects in more detail in Section 2.2.

Our proposed telepresence system is designed to tackle all the aforementioned challenges with a combination of an immersive multi-user VR system with a real-time RGB-D streaming pipeline and two fast, custom point cloud rendering solutions integrated into a state-of-the-art 3D game engine. Our solution enables remote doctors to meet and interact in a virtual operating room with real-time point cloud avatars as well as assist in operations that are live-streamed and visualized in the virtual room in 3D. Our proposed system is capable of handling multiple cameras per location, as our streaming pipeline includes efficient real-time compression and filtering algorithms. Furthermore, we integrated an easy-to-use registration, i.e., extrinsic calibration, method. To evaluate the benefits of VR-telepresence systems in general and ours specifically, we conducted two user studies with doctors exploring possible use cases, comparing the different rendering solutions, and investigating aspects such as spatial and social presence, realism as well as preference. To summarize, our contributions are:

- A multi-user VR-based telepresence system implemented in the state-of-the-art game engine Unreal Engine 4 with a prototype for avatar face reconstruction.
- A modular low-latency multi-camera RGB-D streaming pipeline including filtering, denoising, and compression of RGB-D data, which is easy to extend.
- Custom splat- and mesh-based point cloud rendering solutions and an accompanying user study to compare the two methods.
- An extensive qualitative evaluation of the proposed system as well as a user study exploring clinical benefits and relevant aspects such as spatial and social presence.

### 3.1.2 Related Work

Many VR/AR-based telepresence systems have been proposed in the past, some still rely on video feeds that can be augmented [372, 175], others do make use of real-time point clouds and 3D reconstruction [188]. For instance, Boehlen et al. [25] recently presented a real-time telepresence system intended for usage in caregiving that uses multiple RGB-D cameras as well as point cloud visualization and Anton et al. [9] proposed an augmented telemedicine platform for real-time medical consultation in which the patient is captured via an RGB-D camera, a remote expert can assist using a 3D display, and annotated feedback is sent back to the patient-side. The former system uses 4 Azure Kinect RGB-D cameras that are registered to each other initially using a combination of 3D feature-based coarse registration and ICP-based refinement. Each camera is connected to one compact computer that handles the pre-processing. Eventually, the data gets sent via TCP/IP to the main PC that acts as a client and visualizes the point cloud scene in VR. However, this system doesn't

support multiple users or avatars and doesn't consider compression to reduce the required bandwidth. The mentioned system by Anton et al., in contrast, does employ WebRTC, including the VP8 compression algorithm, for data transmission between the patient's and the remote expert's side. However, the depth data is compressed, too, using the limited 8-bit YUV-based compression. Even though they came up with an adapted approach to encode the depth data into the 8-bit YUV format, accuracy and efficiency are lacking. Moreover, they use only a single RGB-D camera, making the system prone to occlusions. The camera gets registered with the projector for correctly overlaid augmentations using the typical checkerboard approach, while morphological operations are applied to remove noise. At the remote expert's side, the patient/scene gets real-time-reconstructed into a textured mesh. As with the system by Boehlen et al., they do not provide an avatar for the expert. Thoravi Kumaravel et al. [380] successfully demonstrated the benefits of immersive telepresence for remote teaching of physical tasks via a bi-directional mixed reality system that combines AR and VR. In this system, again, multiple RGB-D cameras are used to capture the local user performing a task. The cameras are tracked and registered to each other using Vive trackers, however, the employed method leads to notable offsets. For data streaming, they employ the RoomAlive Toolkit that includes RVL and JPEG compression for depth and color, respectively. Using AR, the local user can see annotations and a simple floating head + hands avatar of the remote user. The remote user, on the other hand, uses VR to observe the local user visualized through a point cloud and make annotations. As the system is symmetric, both users can switch dynamically between AR and VR modes. Point cloud hologlyphs of the remote user can also be viewed in AR, although only with 10 Hz. Gasques et al. [124] also developed a collaborative mixed reality system based on multiple color and depth cameras for live 3D scene capture and reconstruction, and AR and VR devices. We briefly presented the work in Section 2.3. In this work, the marker-based tracking system OptiTrack is used for registration of the cameras, tracking the movements of the people (heads + hands) and objects, as well as to allow for 3D annotations. On the VR side, the remote expert gets the live-streamed 3D patient reconstruction (point cloud), a simple floating-head avatar representing the local doctor, and video live feeds. Using a tracked pen and IMU-equipped gloves, annotations can be made. On the AR side, these annotations as well as a floating head + hands avatar of the expert are visualized to the local doctor. The focus of this system, however, lies more in tracking and interaction, as the depth images are transmitted unprocessed to the remote location. Based on the work by Gasques et al., Roth et al. [311] recently presented another MR teleconsultation system that is intended for telepresence in ICUs. Similar to others, the local user uses AR, and remote ones VR. In their system six Azure Kinect RGB-D cameras are mounted on the ceiling and, as commonly done, connected to dedicated PCs ("capture nodes"). The data is then compressed and transmitted to the remote location where the point cloud is computed but eventually rendered as surface mesh via a custom shader. Remote users – this system supports more than one – are visualized using 5-point tracked full-body avatars and inverse kinematics. The avatars get personalized using a photo of the face taken beforehand and make use of the VR headsets eye tracking for eye animation. The drawbacks they report are the quality of the point cloud, lacking avatar realism, and high latency of 300-400 ms, though.

For telepresence applications, it is of high interest to constrain the required bandwidth to a reasonable level which makes real-time streaming of RGB-D sensor data and point clouds a difficult task. Therefore, one important but often still lacking aspect of such streaming systems is efficient data compression, particularly of depth

data. Not only must the amount of data be reduced as much as possible but also as quickly as possible. Often this is only achieved by sacrificing the quality, which is problematic in medical contexts though. We previously discussed this topic in Section 2.2.5 and explicitly focus on it in Section 3.2. In short, there are two main approaches into which most of the previous dedicated research on RGB-D compression can be split: 2D approaches using image- and video-compression techniques that compress the individual color- and depth images [412] and 3D approaches which directly compress the point cloud. The latter often rely on hierarchical subdivision using octrees [246] and tend to be slower and less effective compared to 2D approaches if the reconstruction quality should remain high or even be lossless [226]. To achieve convincing results, the image- and video-compression techniques need to be adapted to depth images and their specific characteristics though [275]. A comprehensive overview of the recent work in this department was recently provided by Cao et al. [44]. Even with the ongoing work in this field, the problem is far from solved.

Another important task is to produce high-quality 3D visualizations of the RGB-D data. One persistent obstacle is the inherently noisy output of the depth sensors, even the newest ones like the Azure Kinect suffer from temporal noise and effects such as the flying pixel and multipath effects [381]. Since the emergence of low-cost RGB-D cameras, much research was done to enhance the depth images by proposing different denoising, inpainting, and filtering approaches [5, 220, 121]. Though, the problem is still relevant today, as the proposed solutions often have difficulties with dynamic content, e.g., in the form of ghosting artifacts, or take too much time for real-time application. More about this topic can be read in Section 2.2.3 and our contribution dedicated to depth image inpainting in Section 3.3.

To eventually visualize point clouds and, specifically, point cloud and 3D reconstructed avatars, different techniques were proposed. A broad overview of reconstruction and rendering methods is given in Section 2.2.4. While, in principle, very high-quality representations can be achieved via offline scanning and elaborate reconstruction techniques, these methods are not suitable for real-time telepresence applications [235]. Dou et al. [90] presented an online, multi-camera performance capture system with advanced volumetric 3D reconstruction achieving real-time speed, however, multiple high-end PCs were necessary to achieve 30 Hz which leaves no room for other necessary tasks in a telepresence system such as compression, rendering of multiple users, and rendering the scene itself in VR. Similarly, Orts et al. [264] were able to stream high-quality 3D reconstructed avatars in real-time. The avatars are produced via temporal-volumetric fusion of the data of multiple RGB-D cameras, however, the proposed system is computationally highly demanding, requires a 10 Gigabit connection, and is complicated to set up. On the other hand, Gamelin et al. [122] showed that even simpler and faster point cloud visualization techniques such as splatting are sufficient to outperform preconstructed animated avatars in collaborative spatial tasks. Yu et al. [431], too, compared point cloud avatars with pose-tracked (using an Azure Kinect), animated full-body mesh avatars in their telepresence prototype and found the point cloud avatars to be superior regarding perceived co-presence and social presence. Recently, Gunkel et al. [136] presented another RGB-D-based approach for real-time 3D avatars. Using either an Azure Kinect or ZED 2i depth camera, point cloud or mesh avatar representations are computed directly in the shader, based on the depth (and color) images. The proposed pipeline includes various pre-processing and optimization techniques such as background subtraction, hole-filling using morphological operations, edge smoothing, and color encoding for streaming with standard video codecs. However,

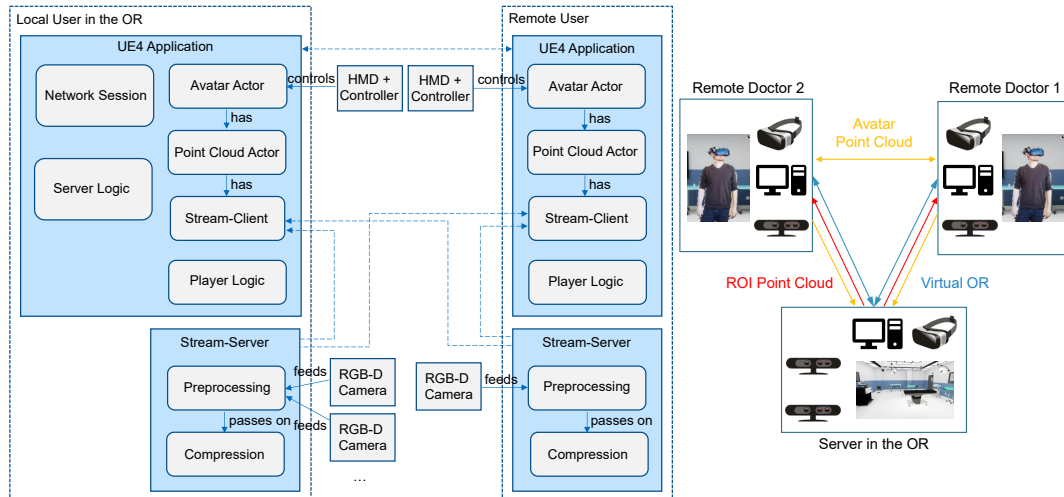


FIGURE 3.1: Left: System architecture of our application. Right: System setup and communication channels between the server in the OR and the remote users.

the resulting motion-to-photon latency is quite high at 300 - 400 ms.

### 3.1.3 System Overview

In this section, we present our telepresence system for remote consultation and collaboration in healthcare. In our system, the doctors and the patient in the physical operating room as well as the remote experts in VR are visualized via accurately registered live-streamed point cloud-based 3D representations. To provide high-quality graphics and robust network components, we decided to use the Unreal Engine 4 as a basis. This also has the benefit that a lot of basic aspects such as collision handling and different HMDs are supported out of the box. A core pillar in our system architecture is the RGB-D streaming pipeline that we realized not with the Unreal Engine, as the engine and its network components specifically are not suited for low-latency transmission of huge data loads. Instead, the RGB-D data is streamed via custom client-server connections that we implemented using C++ and CUDA; we integrated these into our Unreal Engine application. This is illustrated by the system diagram on the left side of Fig. 3.1 which shows our system's components in more detail.

The image on the right side of Fig. 3.1 illustrates the general setup and the communication channels between the participants: The server, which acts as a client too, is located in the operating room, has multiple RGB-D cameras connected to it, and hosts the virtual OR scene. Remote doctors connect to the server and receive the point cloud visualization of the physical scene – the region of interest (ROI) of the OR – from the server. If the users have an RGB-D camera for a personalized point cloud avatar themselves, they directly broadcast the corresponding data to all participants. The application's network traffic, apart from the RGB-D data, however, is always routed via the server.

An example of our telepresence system in action can be seen in Fig 3.2. The left image shows the doctor in the real OR as a live-streamed point cloud visualization in the virtual OR, seen from the viewpoint of a remotely connected doctor. The right image shows the same doctor rendered with the point clouds from two cameras which helps to minimize occlusions. The middle images illustrate the physical





FIGURE 3.2: Example of our system. Left: Live-streamed point cloud visualization of the local doctor in the real OR, seen from the perspective of a remote doctor. Middle: Physical environments of the local (top) and remote (bottom) doctors with RGB-D cameras (see white rectangles). Right: Point cloud visualization of the local doctor using two cameras to minimize occlusions, and part of the remote doctor's self-avatar.

environments of the local and remote doctors with their respective RGB-D cameras.

### Multi-User VR Environment

The central part of our system is a virtual operating room in which all the RGB-D data gets streamed and rendered, and in which the remote doctors can meet using HMDs, see Fig. 3.3 (left). The network architecture is based on Unreal's session system. After starting the application, the users arrive in a lobby where they can start a session or join an existing one. The focus of our system lies in users making use of HMDs to have an immersive VR experience and having real-time personalized point cloud avatars. However, for the case that the required technology is not available, we made sure the system is usable with a mouse and keyboard, too, and integrated flying mesh avatars as a fallback. For VR locomotion the room-scale system is used in which the users can physically walk to move. We also provide a teleport functionality for cases where the physical space runs out. Research showed that this locomotion metaphor exhibits the least risk of inducing cybersickness. Also, to not confuse other present users, a simple particle effect is shown to indicate the deliberate nature of the teleport. Other features of our virtual operating room are a virtual monitor to show medical image data and 3D organ meshes modeled based on real patient data, in this case, livers. The fully synchronized organ models consist of separate parts for arteries, tumors, and a half-transparent outer shell and can be grabbed and inspected by the users, as can be seen in Fig. 3.3 (right). Also, the organ data is quickly replaceable to represent new cases.

### Point Cloud Streaming

In this section, we describe the point cloud streaming pipeline of our telepresence system, which is one of its core parts. Instead of implementing everything from scratch, which would be tedious and time-consuming, we opted to use "DynCam: A Reactive Multithreaded Pipeline Library for 3D Telepresence in VR" by Schröder



FIGURE 3.3: Our virtual operating room with a point cloud avatar of a remote user on the left and an interactive virtual liver on the right.

et al. [325] as a base and extended it for our needs. Generally, the Dyncam library provides a good foundation, as it has low latency streaming capabilities, which are crucial for telepresence in VR, and is easily extendable. However, we found some aspects such as compression and rendering to be lacking and decided to extend or replace them. We also adapted the architecture so that a single server-client connection can handle multiple cameras to reduce overhead and integrated functionality to record point cloud sequences and replay them later without the need for cameras to be connected. In Fig. 3.4 you can see an illustration of our final streaming pipeline. One or multiple RGB-D cameras are connected to the streaming server and will be processed individually. We use the new Azure Kinect RGB-D camera from Microsoft, as it has a high resolution and precision, hardware synchronization, and uses the time-of-flight (TOF) principle which is very suited for indoor use. The first step of our pipeline is the preprocessing of the color and depth images, which consists of lens correction, cropping, and filtering algorithms including background subtraction, and morphological filters for hole-filling and denoising. Then the images get compressed and transmitted to the client where they will be decompressed. We decided to compress and transmit the color- and depth images instead of point clouds, as this enables us to use more efficient image- and video-based compression algorithms. This means that the point cloud will only be computed client-side. For this, the camera's intrinsic data will be transmitted once too. Lastly, the point clouds of multiple cameras will be registered to each other and to the virtual scene and then rendered.

We have implemented the filtering algorithms using CUDA to minimize latency. For background subtraction, there are two options. The first one is to set a simple depth threshold per camera to exclude points from rendering (depth set to zero). As a second option, the user can do a one-time recording of the scene, e.g., the empty operating room. In this case, the pixel-wise minimum valid value will be stored and acts as the threshold to distinguish between foreground and background. Algorithm 1 describes the whole preprocessing pipeline in more detail. For hole-filling and depth denoising, we did extensive experiments with various filter algorithms, such as optical flow, Navier-Stokes-based inpainting, local regression, etc, but found them to be too slow, or the provided improvements were not significant enough to warrant the additional performance cost. Thus, we settled for faster solutions that achieve a good trade-off in this regard. Hole filling is done via multiple iterations of median-based morphological filtering (see Alg. 2) and denoising using an adapted Kalman filter. As the Azure Kinect camera masks off the corners of the depth images with non-usable data, we added the option to do a cropped transmission and

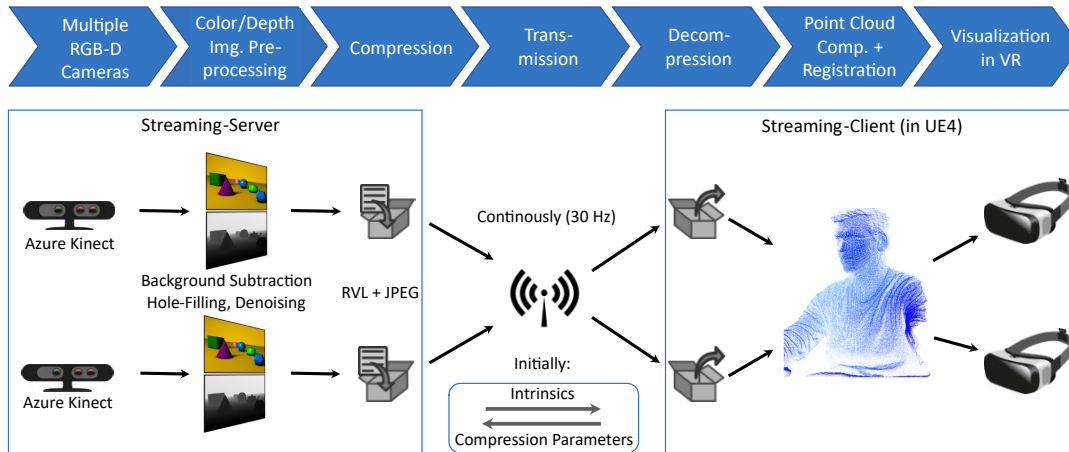


FIGURE 3.4: Our point cloud streaming pipeline. Color and depth images of multiple cameras get individually preprocessed and compressed server-side and then transmitted to the clients where the point clouds get computed and registered before rendering.

save bandwidth. Regarding the compression of the depth images, we extend the solution present in Dyncam (quantization plus LZ4) by integrating the H.264 video codec, and multiple efficient lossless algorithms, i.e., an ANS coder, RVL, and Z-standard. We found that even after compression the depth images are responsible for most network traffic while the color images are sufficiently small by just using JPEG compression. Therefore, we adapted the integrated lossless depth compression algorithms to achieve higher compression ratios by adding temporal delta compression. As with all compression and streaming systems, at some point, a trade-off has to be made between the required bandwidth and the computational speed. To allow the users to adapt this to their local capabilities, i.e., hardware power and network bandwidth, our design allows the client-side user to select the compression algorithm and parameters that will be used for their individual connection. For example, one user could choose to use RVL compression, which is fast and efficient, while another user may have a slower internet connection and opt for a different compression technique with a higher compression ratio at the cost of increased computational costs and lower speeds. The modular design of our pipeline also makes it easy to implement other and even more efficient compression algorithms in the future.

### Point Cloud Registration and Rendering

To be able to have individual point cloud avatars for remote users, a registration procedure between the RGB-D camera and the VR coordinate system has to be done. When using multiple cameras, these have to be registered to each other too. For these registration tasks, we use the novel method by Mühlenbrock et al. [248]. It uses a grid-like registration target that is visible in the depth images to register multiple RGB-D cameras with each other and the VR coordinate system. This registration method proved to be very quick and easy to use. In the virtual scene, we have a hierarchy of actors in which each one is responsible for rendering one point cloud/camera. To account for the registration occasionally being slightly off, we allow the user to manually tweak the transformation in-game via sliders.

---

**Algorithm 1** Depth Preprocessing

---

**Require:** captured color and depth images, device calibration**if not** *Init* **then**    *Init*  $\leftarrow$  *True*    *Transf*  $\leftarrow$  *ComputeUndistortTransf*(*DeviceCalib*, *Alpha*)    **if** *BackgroundRecording* available **then**        *BackgroundImg*  $\leftarrow$  *ComputePixelwiseMin*(*BackgroundRecording*)    *DepthImg*  $\leftarrow$  *ApplyRangeFilter*(*DepthImg*, *MinRange*, *MaxRange*)    **if** *BackgroundImg* available **then**        *DepthImg*  $\leftarrow$  *ApplyBGSubst*(*DepthImg*, *BackgroundImg*)    *DepthImg*  $\leftarrow$  *ApplySpatioTemporalFilter*(*DepthImg*, *FilterID*, *Sensitivity*)    *DepthImgT*  $\leftarrow$  *Undistort*(*DepthImg*, *Transf*)    *DepthImgT*  $\leftarrow$  *Crop*(*DepthImgT*, *Radius*)    *RegColorImg*  $\leftarrow$  *TransfColor2Depth*(*DepthImg*, *ColorImg*, *DeviceCalib*)    *RegColorImgT*  $\leftarrow$  *Undistort*(*RegColorImg*, *Transf*)    *RegColorImgT*  $\leftarrow$  *Crop*(*RegColorImgT*, *Radius*)

---

---

**Algorithm 2** Hole-Filling & Outlier Removal

---

**Require:** captured depth image**for** Pixel *P* in *DepthImg* **do** ▷ outlier removal    *Neighbours*[]  $\leftarrow$  *ComputeNeighbours*(*P*)    *Sum*  $\leftarrow$  *ComputeSum*(*Neighbours*)    **if** *Sum* < *Threshold* **then** ▷ default: 10 mm        *P*  $\leftarrow$  0**for** *it* in *iterations* **do** ▷ hole-filling; e.g., 3 iterations    **for** Pixel *P* in *DepthImg* **do**        **if** *P* < *Threshold* **then** ▷ default:10 mm            *Kernel*  $\leftarrow$  *ComputeKernel*(*Radius*)            **for** *Value*, *i* in *Kernel* **do**                *KernelValues*[*i*]  $\leftarrow$  *Value*            *KernelValues*  $\leftarrow$  *Sort*(*KernelValues*) ▷ e.g., using bubble sort            **if** *KernelValues*[*KernelSize* - 1]  $\geq$  *Threshold* **then**                *Zeroes*  $\leftarrow$  *CountZeroes*(*KernelValues*, *Threshold*)                *P*  $\leftarrow$  (*KernelSize* + *Zeroes*)/2 ▷ median of valid values

---

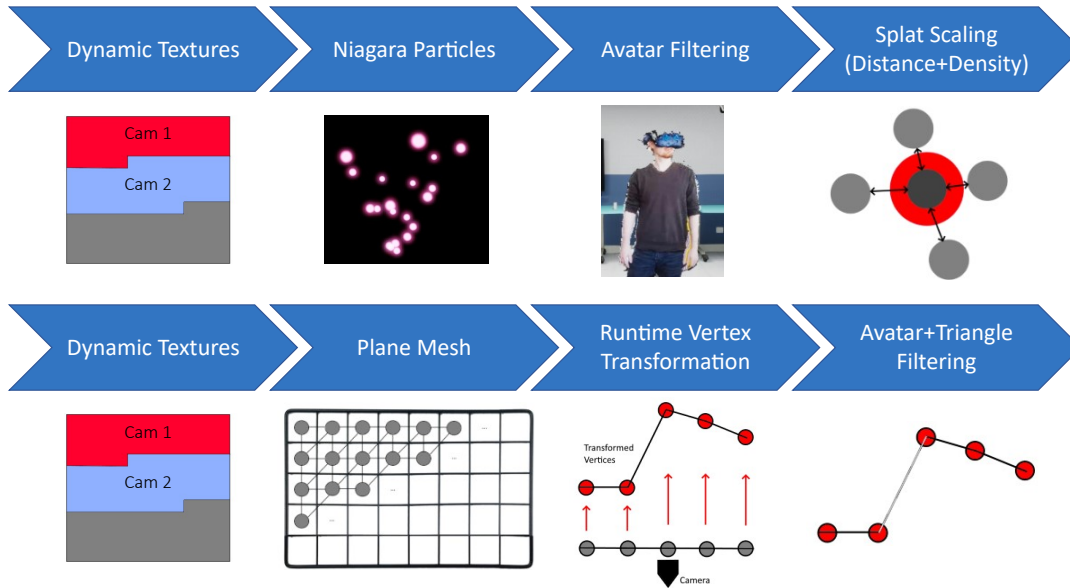


FIGURE 3.5: Pipelines of our two rendering solutions: one is splating-based (top) and the other one mesh-based (bottom).

Of paramount interest is the fast and visually pleasing rendering of the point clouds. The Unreal Engine historically does not natively support point cloud rendering, however, by now, there is a publicly available point cloud rendering plugin “LiDAR Point Cloud”<sup>1</sup>. We experimented with the plugin but found it to be too slow with dynamic point clouds and, therefore, not suitable for our application. Our investigations indicate that the reason for the poor performance is that the plugin was designed to handle huge but static point clouds – it builds a spatial acceleration data structure intended for level of detail (LOD). With dynamic point clouds, this costly operation would have to be done in each frame. The rendering solution provided in Dyncam was not convincing to us, both visually and from a performance point of view. We also considered implementing more complex volumetric reconstruction techniques similar to the ones in [264] and [90] but eventually decided against it, as they are computationally highly demanding, and we prioritized keeping the latency, which is critical in VR, to a minimum. Therefore, we have developed two different and very quick rendering solutions that we both integrated and tested in the Unreal Engine.

The first method is splating-based but uses Unreal’s new Niagara particle system, see the top part of Fig. 3.5. To get the point cloud positions and colors from the CPU to the GPU, we adopted Dyncam’s approach of using two dynamic textures. In our case, all cameras from one user share a single texture with the size of  $2048^2$  which is sufficient for at least 11 cameras. Via Niagara module scripts we then calculate the UV coordinates based on the particle ID and forwarded parameters such as the point count per camera, texture size, etc., exploiting the fact that the point clouds are ordered. For point clouds meant to represent avatars of VR users, we additionally filter out all points that exceed a set distance from the center of mass which we compute dynamically via the HMD position. Also, points representing the HMD are filtered out for the local avatar’s user in a similar fashion to prevent occluding the vision. When using splating methods, the size of the points is important. To minimize holes and overlaps, we compute the diameter of each point based on the

<sup>1</sup><https://www.unrealengine.com/marketplace/en-US/product/lidar-point-cloud>

distance recorded from the sensor, as, because of the parallel projection, the density of points decreases with the distance. Another effect to consider is that surfaces perpendicular to the line of sight of the camera get sampled with a significantly lower density which results in bigger holes. To account for this fact, we also dynamically compute the local density of points, which is computationally cheaper than approximating their normals, and adjust the diameter accordingly. As blend mode in the material we use the masked mode instead of the translucent mode to circumvent the costly depth sorting.

Our second rendering method is based on a very fast mesh reconstruction (see bottom part of Fig. 3.5) and is intended to prevent visible holes between individual points altogether and instead provide continuous surfaces. With our mesh reconstruction method, we can again exploit the fact that the point cloud is ordered and establish a one-to-one relation between point cloud points and the vertices of a plane mesh. At start-up, we once automatically create a plane mesh with the exact vertex dimensions and structure as the depth image on which the point cloud is based. For example, a rectangular grid-like pattern of  $640 \times 576$  vertices. At runtime, we then make the point cloud positions and colors available to the mesh's material, again, via dynamically updated textures. In the material, the vertices get transformed via Unreal's `WorldPositionOffset`-function according to the corresponding point cloud positions in the RGB-D camera's local space and the world-transform. We wrote custom shader nodes to exclude triangles from being rendered (alpha set to zero) if, based on the original position, the point was invalid, or one of the triangle's edges is too long. This prevents long, stretched triangles between foreground and background objects. Fig. 3.6 shows a comparison of the two renderers. As can be seen, in some instances the individual points are still clearly visible with the splatting method.

### 3.1.4 Results

#### Performance

To quantitatively evaluate our system's performance, we measured the time needed for filtering, compression, and rendering as well as the frame rate with which camera updates can be processed. All performance measurements were done using a PC with Windows 10, an Intel i7 7800x processor, 16 GB of main memory, and an Nvidia GeForce 2070 graphics card. As HMD we used the HTC Vive Pro Eye with a mounted Facial Tracker. Our application was developed using the Unreal Engine 4.26. All measurements were done without background subtraction to maintain the full worst-case workload.

First, we evaluated the speed of the filtering step of our pipeline (see the top left part of Table 3.1). As a whole, it took 1.34 ms to filter an un-cropped depth image ( $640 \times 576$  pixel), of which 0.59 ms were spent on hole-filling. Next, we measured the time for compression and decompression of the registered color image using JPEG: each took 1.5 ms. Using the cropped transmission ( $540 \times 476$  pixel) to discard the Azure Kinect camera's unused border areas resulted in only 1.08/1.22 ms being needed. Using H.264 (preset: ultrafast, tune: zerolatency), the computation was significantly slower: 7.72 ms for compression and 4.84 ms for decompression of the full-sized color images, and 11.04 ms and 9.4 ms for the depth images, respectively. Compression and decompression of the un-cropped depth images using the RVL method, which we found to be the most efficient, took 1.76 ms and 1.19 ms, respectively. Cropping reduced the time needed to 1.31 ms and 0.987 ms, respectively. An



FIGURE 3.6: Comparison of the two point cloud renderers, splatting in the top, fast mesh reconstruction in the middle. Both look quite good but the splatting method still has visible holes in some areas, see the red rectangles. With the mesh, however, object borders can look jagged (see highlighted region). The bottom image shows the captured scene.

TABLE 3.1: Performance measurements of our application.

Task Duration	Time (ms)	
	Full	Cropped
Filtering	1.34	-
Color Comp. (JPEG)	1.50	1.08
Color Decomp. (JPEG)	1.50	1.22
Color Comp. (H.264)	7.72	-
Color Decomp. (H.264)	4.84	-
Depth Comp. (tRVL)	1.76	1.31
Depth Decomp. (tRVL)	1.19	0.99
Depth Comp. (H.264)	11.04	-
Depth Decomp. (H.264)	9.40	-
Latency		
VR Round-Trip Time	22-29	-
PCloud MotionToPhoton	120-150	-
Compression		
	Image Size (kB)	
Color (JPEG)	21.9	18.0
Color (H.264)	15.9	-
Depth (RVL)	208.0	-
Depth (tRVL)	96.6	81.8
Depth (H.264)	59.7	-
Rendering Perf. (ms)		
	PCloud	Mesh
VR Frametime (1 Cam)	10.9	8.5
VR Frametime (2 Cam)	15.0	13.0

important detail to note is that the (de-)compression of color- and depth are done in parallel, meaning that the time won't stack on top of each other. The results show that in our pipeline, preprocessing and compression are very fast and can be accelerated even further by cropping the unused borders of the Azure Kinect's depth images, although the speed-up doesn't reach its theoretical potential (19-28 % less time for 30 % fewer pixels).

After these individual measurements, we evaluated the overall performance by measuring the more comprehensive metrics of the point cloud update rate that was maintainable as well as the eventual fps/frame times in VR. We measured the point cloud update rate by calculating the delta time both in our streaming server application as well as in the UE4 telepresence application that received and rendered the data. Throughout all cases, even using two cameras sending in full resolution and being in VR at the same time, our system was able to maintain a delta time of 33 ms which corresponds to the 30 fps capturing capability of the Azure Kinect cameras. The final performance in the packaged VR application was not only dependent on the number of cameras and the rendering technique but also heavily dependent on the general graphics settings the scene was rendered with ("scalability settings" in Unreal). Using the mesh rendering technique, Unreal Engine's "high" graphics preset, and the full depth resolution, we achieved a frame time of 12.5 ms with one camera and 15.5 ms with two. Setting the graphics preset to "low", we were able to reach 8.5 ms and 13 ms, see the bottom right part of Table 3.1. We found the splatting technique to be slower than the mesh variant, achieving only 10.9 ms and 15 ms under the "low" graphics preset. Important to note here is that the rendering resolution was held constant throughout the presets, and the graphics preset had no



effect on the visualization of the point cloud rendering but only on the surrounding scene, most noticeably on reflective materials, anti-aliasing, and ambient occlusion. As can be seen from the performance measurements, our pipeline is very efficient throughout all stages and enables VR performance even with just a single PC per location. Both of the rendering techniques are very quick to compute, but especially the mesh version is highly efficient. As the performance scales with the number of cameras, at some point (e.g., 4+ cameras), more powerful hardware or a second PC would be needed to maintain the real-time VR performance, though.

## Network

Regarding network performance metrics, we measured the round-trip time for interactions in VR, and the motion to photon latency of the whole pipeline, see the bottom left part of Table 3.1. The round-trip time – the time it takes for a client-side action to be transmitted to the server and back to a client – was between 22 ms and 29 ms., depending on the tick rate the server and client could achieve. The tests were conducted with 2 PCs in a local network. With greater distances, e.g., different cities, the time will likely be higher. The time between the camera capturing the scene and it being rendered on the display – the motion to photon latency – was measured by pointing a camera in such a way that both the physical scene and the display were recorded by an external camera. By analyzing the videos frame-by-frame, we found a latency of 4 to 5 frames which corresponds to a 120-150 ms delay. However, roughly half of this is caused by the camera itself, as it is reported that delivery of the raw images by the Azure Kinect software development kit (SDK) needs  $\sim 75$  ms, depending on various parameters. We couldn't find any significant differences in delay for a varying number of cameras, between the rendering methods, or different graphics presets, which may also be because the external camera that we used for the measurement itself only captured with 30 Hz. Although the measurements were not highly precise due to the limited temporal resolution of the external camera, the results show a rather low delay for such a system.

Lastly, we measured the compression efficacy and bandwidth required to transmit the RGB-D Data. The color images with an original size of 1440 kB were compressed with JPEG to 21.9 kB (on average) with a compression ratio of 66. Cropping further reduced the size to 18 kB, as can be seen in the top right part of Table 3.1. With H.264, the color images were compressed to 15.85 kB, and the depth images to 59.69 kB. However, the size and image accuracy are heavily dependent on the parameters; we used constant rate factors (CRF) of 20 and 10 for the color and depth images, respectively. For depth images, we found the RVL algorithm to be a good trade-off between speed and compression ratio. Using it, depth images were losslessly compressed from the original 720 kB down to 208 kB with a compression rate of 3.46. With our temporally extended RVL, the average compressed size shrunk to 96.6 kB with a compression rate of 7.45, though the achievable compression here strongly depends on the amount of motion in the scene. With the cropped transmission, the size was further reduced to 81.78 kB. With the 30 images per second that the cameras provide, our system requires per camera 3,555 kB/s in full and 2,993 kB/s in cropped mode, which corresponds to 23.4 and 27.8 Mbps. This is a very good result considering that the depth images are transmitted losslessly and the high computational speed the compression runs on. Naturally, using lossy compression algorithms such as H.264, the bandwidth could easily be reduced further at cost of higher latency, if need be. However, we noticed visible artifacts on H.264-encoded depth images.



FIGURE 3.7: Our telepresence system in action during the studies: on the left the remote doctor and on the right the live-captured operating room scene.

### 3.1.5 User Studies

To demonstrate the capabilities of our telepresence system and evaluate it regarding crucial aspects such as visualization quality, realism, and spatial- and social presence, we conducted two user studies with doctors and medical students in a hospital.

#### Study 1: Qualitative Feedback, Presence, and Preference

The goal of the first user study was to get general feedback from the doctors, evaluate it regarding relevant aspects like presence, and see how beneficial the telepresence system could be in clinical practice. The study was conducted with  $N_1 = 12$  doctors and medical students with varying amounts of experience in the operating room and with AR/VR. The experimental setup for this study was as follows: In a room similar to an operating room, we divided the space in half. In one half, a PC was set up that acted as the server for the streaming pipeline and host for the VR session. Connected to the PC was an RGB-D camera facing an operating table and a staff member acting as a surgeon, see Fig. 3.7 (right). In the second half of the room was a second PC with an HMD which the participants had to put on and join the session in the virtual operating room where they could see the operating table and staff member as a live-streamed point cloud, see Fig. 3.7 (left). The PCs were connected via a local network.

The task for the participants was to observe the staff member and help him with specific spatial tasks he had to perform with interlocking bricks (Lego). We did use interlocking bricks for this study, as their shape, size, and inherent ability to be combined into various more complex structures makes them suitable to recreate spatial tasks done by surgeons. First of all, the staff member did work alone so the participants could familiarize themselves with the VR experience and the scene. After roughly one minute, the staff member started asking questions and presenting problems to facilitate interaction with the participant and steer his attention to individual bricks. Questions were, for example, how many bricks of one color were used in a construction, how many studs of one color were visible, or how a specific construction could be built with a set amount of available bricks. After the VR experience, which lasted roughly 8 minutes, the participants had to fill out a questionnaire. The questionnaire consisted of a demographical part (age group, sex, experience

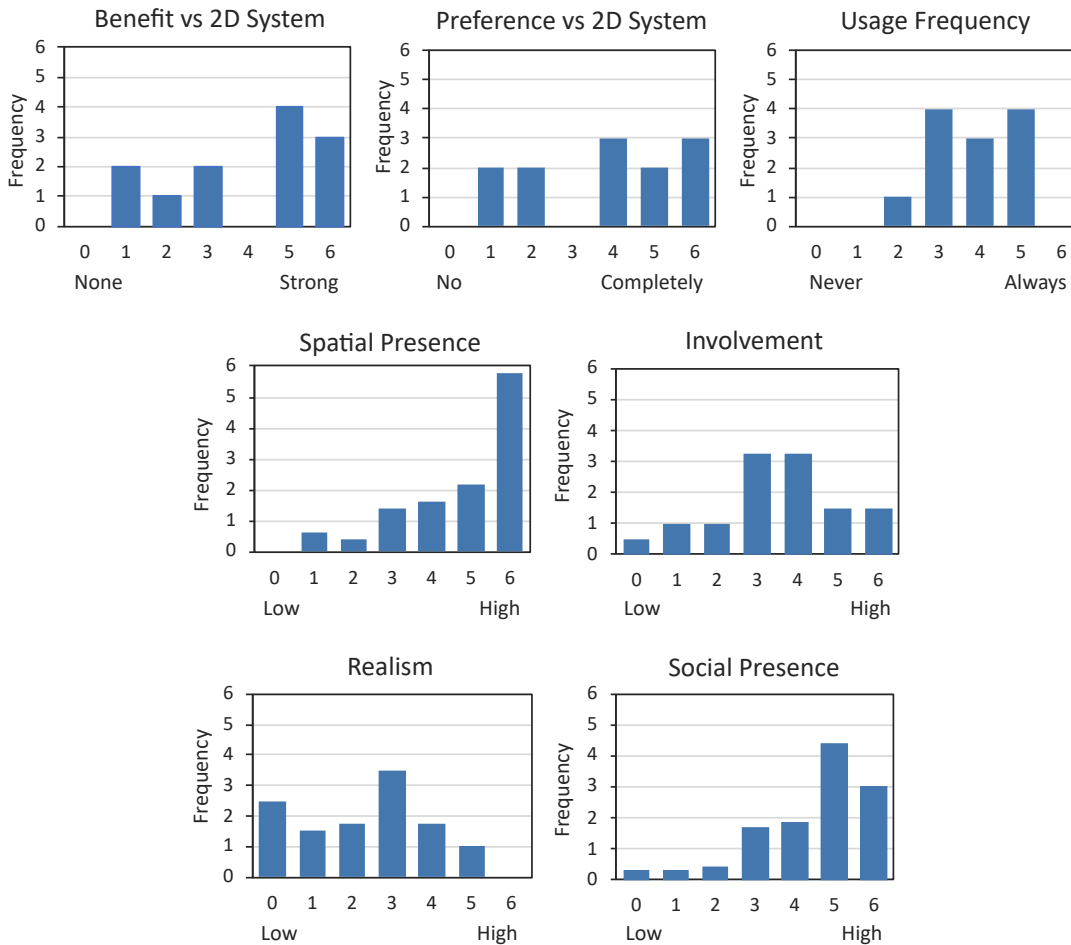


FIGURE 3.8: Results of our first user study (in Likert scores, higher scores are better). Especially the spatial- and social presence scored very well, and involvement and realism were mediocre. The users see moderate to high benefits of the system and most of them would like to use it.

in the operating room, experience with AR/VR), the Igroup Presence Questionnaire (IPQ) [327], which is split into the three subscales spatial presence, involvement, and realism, and a social-presence part taken from Nowak et al. [261]. Additionally, we added various specific questions: about cybersickness; if the participants see benefits of our system compared to more traditional video-based systems; if they would prefer it to those systems; how often they would want to use our system. For all questions, we used a 7-point Likert scale (1-7, but shifted to 0-6 for the evaluation; higher scores are better).

The results of this study can be seen in Fig. 3.8. Note that the scores can be fractional in some cases, as the IPQ subscales are aggregations of multiple questions. Our system scores especially high regarding both spatial- and social presence. 58 % of the participants stated that they had a strong feeling of being present in the virtual world (Likert scale  $\geq 5$ ), the most often given score even being the maximal one. For the rest, the feeling was still moderately pronounced. Similarly, 62 % stated that they had a strong feeling of actually being in the same room with the other person/having a personal meeting with another real person. Only 5 % definitely had not the feeling. We also found that our system fares very well with cybersickness, as no participant had a significant occurrence of it, and 75 % had nearly none to none.

The results for the involvement and realism subscales of the IPQ are moderately good, most participants gave scores relating to “somewhat captivated by the virtual world” or “moderately real world”, although, especially on the question “The virtual world seemed more realistic than the real world” of the IPQ, 75 % gave the minimal score dragging down the subscale. One possibility for this particularly low result could be that the participants took the question too literally, and the question may be not that appropriate in our context. The other questions of the realism subscale<sup>1</sup> scored significantly better. In the end, 25 % of the participants would attest moderate advantages and 58 % even strong advantages to our system compared to more traditional videoconferencing systems. Also, 83 % would prefer this system at least somewhat over teleconferencing systems, and 33 % would use it on all possible occasions, while the other 77 % at least sometimes. The results for the realism subscale as a whole are in line with the comments made by some participants during and after the study – that the point cloud visualization is still somewhat grainy and low precision.

### Study 2: Comparison of Point Cloud Rendering Solutions

In a second study, we compared our two point cloud rendering methods and further investigated the specific clinical use cases in which our system could provide benefits. The experimental setup was similar to the one in the first study, but this time there were two VR phases for the participants. First, they saw the live-streamed staff member with the interlocking bricks using one rendering method, filled out a questionnaire, and then repeated the procedure with the second rendering method and a second part of the questionnaire. Which rendering method was seen first, point cloud or mesh, was randomized. The task in both phases was the same as in the first study, helping the staff member with the interlocking bricks. For this study, we discarded the social presence part from the first study and the spatial presence and involvement subscales of the IPQ, retaining only the realism one. Instead, we focused on getting more precise feedback regarding the potential benefits and use cases of the system. We also directly asked which rendering method would be preferred. The study was conducted with  $N_2 = 7$  doctors and medical students in a hospital.

The results can be seen in Fig. 3.9. The top diagram shows the answers to the question of which rendering technique the doctors would prefer; the value of 3 means they found them similar, lower values mean the splatting technique was preferred, and higher values correspond to the mesh visualization. The bottom diagram shows the scores of the realism subscale for the two rendering techniques. Again, scores are fractional, as the IPQ subscale is an aggregation of multiple questions. The results show that there was no absolute preference for one rendering method or the other. According to the data, the participants found them to be rather similar, some slightly preferring the splatting and others the mesh method. From the direct question and its result in the left diagram, we can see a small tendency for the mesh technique, which lines up with verbally given statements from two doctors after the study that they found the mesh rendering a bit better. The results for the realism subscale also indicate that there is no clear advantage for one or the other method. With a sample size of seven, it is hard to make statements with any significance, though. We did perform a Wilcoxon rank-sum test to test the null hypothesis of both distributions being equal and had not enough evidence to reject this hypothesis. Similar to the first study, the question about if the virtual world was more

<sup>1</sup><http://www.igroup.org/pq/ipq/download.php>

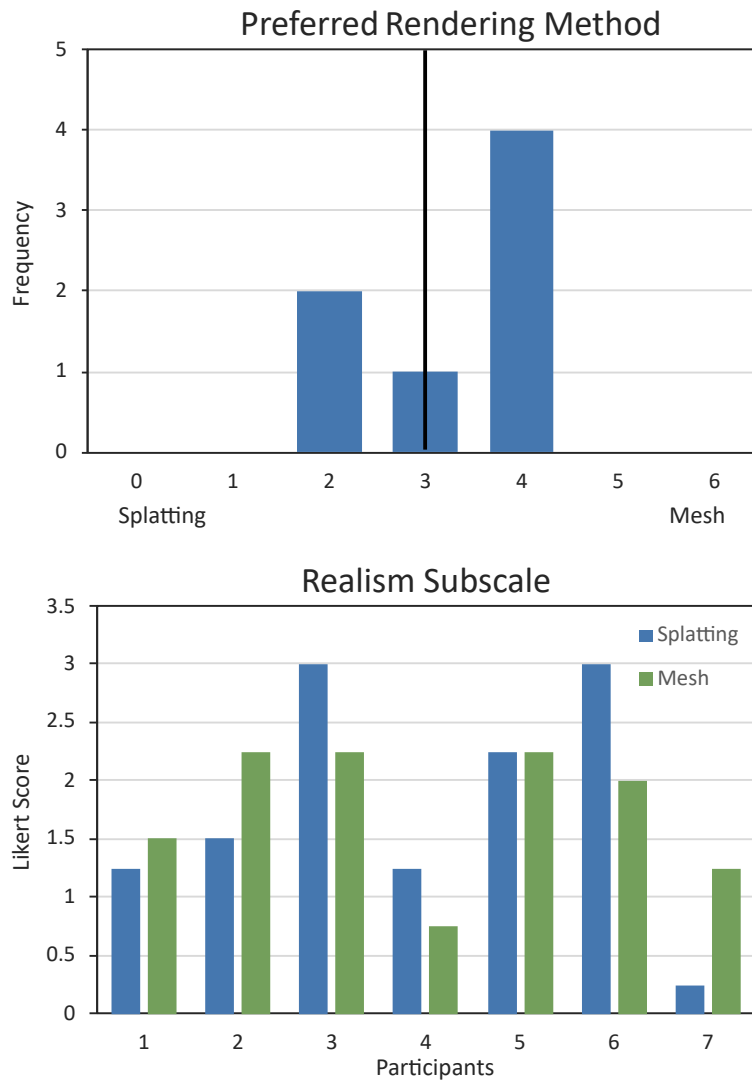


FIGURE 3.9: Results of our second study in which we compare the two rendering methods. The top diagram illustrates that the users have no definitive preference, some slightly prefer the splatting (left side of the x-axis), and others the mesh rendering (right side of the x-axis). The bottom diagram shows that also the realism scores fairly similar between the two rendering methods (higher scores are better).

realistic than the real world scored particularly badly. Asked about it, one doctor argued that a positive answer to this would be impossible, as he obviously knew that he was in a virtual world during the experiment. The rest of the questionnaire's answers confirmed the results of the first study: most doctors would like to use such a telepresence system from time to time (57.1 %), and some would even be eager to use it very often (28.5 %). The more in-depth questions about the benefits and use cases showed that most doctors see moderate benefits over video-based solutions for our proposed system in its current state and very strong benefits if, in the near future, the point cloud visualization quality and precision could be improved. The doctors stated that the system could be advantageous and helpful in emergency operations, or if an inexperienced doctor is on duty. Additional use cases given by the doctors were educational operations, learning of the anatomy, patient anamnesis, and learning and teaching of practical skills in general.

### 3.1.6 Limitations

From the results of the first and particularly, the second study, we conclude that our telepresence system is a very good basis and has very high potential, but the RGB-D cameras' sensor resolution is a main limiting factor at the moment, at least concerning tasks involving high precision or small details. Naturally, sensors with higher resolutions being released would bring the biggest relief, however, we also think about combining 3D cameras with different sensing techniques to complement each other. For example, stereo cameras, which typically produce higher-resolution color and depth images, could be added and used to enhance the fidelity of our system. Another limitation of our current system is the lack of a dedicated point cloud/mesh fusion process. Individually rendering multiple point clouds or meshes that depict the same object leads to visible seams or artifacts, even though they are registered precisely. The flying pixel effect and internal interpolation routines in the cameras may be one of the causes. Efficient and precise real-time fusion is a challenging topic in itself though, for example, Dou. et al. [90] proposed a sophisticated but computationally demanding approach, and, thus, was not the focus of this work.

### Face Reconstruction

A glaring problem with real-time reconstructed avatars in VR/AR telepresence applications is the HMD blocking the face. To be able to see the people's faces is highly important in collaborative virtual environments, though. To solve this issue, we developed a prototype combining the HMD's eye- and mouth-trackers with 3D Morphable Face Models, see Fig. 3.10 for an overview of the whole pipeline.

A straightforward solution would be to use the deformable face model delivered with the HTC eye-tracking SDK and let the trackers drive the deformation. However, this generic model wouldn't be too realistic, as it can't be personalized. Also, we found the resulting facial expressions often do not match the actual mimic that well, sometimes even being weird-looking. Therefore, we acquired a more advanced morphable face model from "eos: A lightweight header-only 3D Morphable Face Model fitting library in modern C++11/14" [148] which can be automatically adapted to the person's facial geometry via a photo taken beforehand. The library actually includes two face models, the Surrey Face Model and the 4D Face Model, the latter covers not only the face but the whole head. However, empiric tests that we conducted showed that the Surrey Face Model lead to better results, which is why we decided to employ it. To fit the photo to the facial geometry, facial landmarks

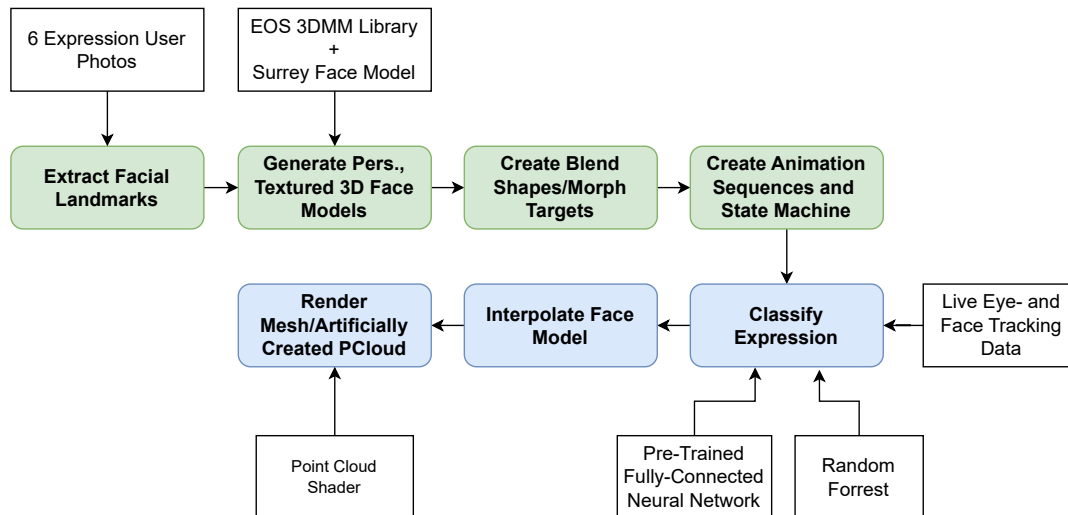


FIGURE 3.10: Depiction of the face reconstruction pipeline. Steps in green only have to be done once in pre-processing, steps in blue are continuously repeated in real time.

have to be extracted from the image first, see Fig. 3.11 (top row) for an example of the fitting process. For this, we used a slightly adapted version of the facial landmark detector implemented in the Dlib library that generates 68 point landmarks.

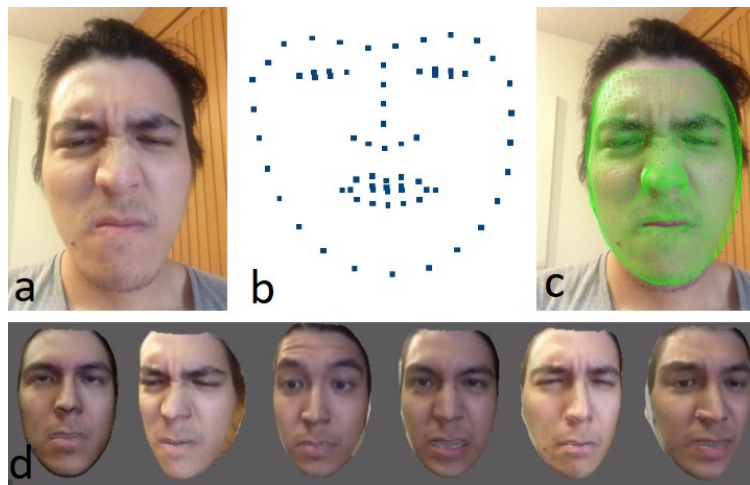


FIGURE 3.11: Top row: fitting a morphable face model (c) to a photo (a) using extracted facial landmarks (b) and the eos library. Bottom row/(d): Six facial expressions applied to the face model.

To further customize the face model, we do not simply directly apply the trackers' output to the morphable face but pre-animate six facial expressions relating to basic emotions that are dynamically selected, interpolated, and applied to the face. For this, we initially take six photos of the user's face, one for each of the facial expressions, and generate a fitted face model each. Following this, we create a unified model with blend shapes/morph targets for all the expressions, and then generate animation sequences and a small state machine to drive the animation and interpolation of the geometry and textures. Instead of a fully-connected state machine that is able to directly interpolate between all facial expressions, we decided to use a centralized one in which animations always transition over the neutral expression



FIGURE 3.12: Our mesh-rendered point cloud avatar with the reconstructed face; on the left with a neutral look, and on the right showing disgust.

to reduce the complexity. The selection of the facial expression is based on either a custom (fully-connected) neural network that we pre-trained to map the trackers' output to the emotions, or a random forest classifier. The six basic emotions we decided on are sadness, disgust, happiness, surprise, anger, and neutrality. Fig. 3.11 (bottom row) shows the six expressions applied on the face model. To train the neural network, we created a small data set of facial expressions by asking multiple people to mimic the six emotions while we record the tracker output. The morphable face model is eventually rendered at the HMDs' 3D position while the original point cloud points or mesh triangles corresponding to the face are hidden. An example is illustrated in Fig. 3.12. To get a homogeneous avatar appearance for the point cloud rendering method, we wrote a custom shader transforming the mesh of the morphable face to look like a point cloud.

### 3.1.7 Conclusions and Future Work

With this work, we have presented an immersive VR telepresence system for telementoring and remote collaboration in the operating room. Multiple Azure Kinect RGB-D cameras and point cloud avatars enable doctors to interact with each other and to view and assist in operations from a distance as if they were there. Thanks to our modular, low latency RGB-D streaming pipeline and efficient point cloud rendering and reconstruction techniques implemented in the Unreal Engine 4, we achieve very low latencies: our evaluation shows motion-to-photon latencies of only 120-150 ms, of which half of the latency is caused by the camera itself. At the same time, our pipeline handles all relevant tasks from filtering, denoising, and compression of the RGB-D data, to registration and computation of the point clouds. Using lossless compression, a bandwidth of only 23.4 Mbit/s per camera is required, although this can be further reduced by lossy compression when appropriate. In contrast to many other telepresence systems, our proposed solution is easy to set up and doesn't require multiple high-end PCs to run. We also presented a prototype to tackle the issue of occluded faces via personalized semi-real-time face reconstruction. A user study we conducted with doctors indicated that our system is capable of inducing very strong feelings of spatial and social presence. 83 % of the participating doctors attested moderate to high benefits to our system compared to video-based solutions, and one-third would like to use it at every opportunity. Lastly, we compared two point cloud rendering solutions, splatting and fast mesh reconstruction, via another study. The results showed that they scored quite similarly regarding the



IPQ's realism subscale, and the doctors had no clear preference for one or the other method. If directly asked for a comparison, there was a slight tendency in favor of the mesh method. In general, though, the RGB-D sensors' lacking resolution seems to be a limiting factor. Accordingly, most doctors would attest medium-high benefits to the system in its current state, reaching from educational scenarios to consultation in emergency situations, and very strong benefits if the fidelity could be improved upon in the near future.

Possible work for the future would be to enhance the rendering quality by improving both the real-time preprocessing of the RGB-D data as well as the rendering solutions themselves. For instance, neural networks became popular in many related fields lately and may lead to improvements in hole filling, denoising, or merging of RGB-D data and point clouds, too. In order to further increase the fidelity, one could add dedicated color or stereo cameras. Deep-learning-based up-sampling of the depth images could also be promising. Another aspect that could be further improved upon is the compression of the depth data. State-of-the-art video compression algorithms may be adapted to better suit the compression of depth images, however, they are computationally expensive and not necessarily lossless which is crucial to not lose information from medically important areas/the situs. Using different compression algorithms and a combination of lossy and lossless techniques depending on the area of the scene and the visualized object may be a valid solution to circumvent this problem. To tackle the issue of the obstructed faces, the prototype of our proposed reconstruction pipeline should be finalized. Point cloud/mesh fusion is also an important topic we would recommend to explore. Lastly, integrating the option to also use AR HMDs would be a great addition, especially for the doctors in the operating room.

## 3.2 (Improved) Lossless Depth Image Compression Methods

Since RGB-D sensors became massively popular and are used in a wide range of applications, including especially VR telepresence systems such as the one presented by us in the previous Section 3.1, depth data compression became an important research topic. Live-streaming of depth data requires quick compression and decompression. Moreover, accurate preservation of information is crucial in order to prevent geometric distortions. As mentioned in Section 1.1, custom algorithms are needed considering the unique characteristics of depth images.

Therefore, in this section, we propose a real-time, lossless algorithm which can achieve significantly higher compression ratios than RVL. The core elements are an adaptive span-wise intra-image prediction and parallelization. Additionally, we extend the algorithm by inter-frame difference computation and evaluate the performance regarding different conditions. Lastly, the compression ratio can be further increased by a second encoder, circumventing the lower limit of four-bit per valid pixel of the original RVL algorithm.

The work presented in this section is based on our published paper PC5 in Appendix A.

### 3.2.1 Introduction

Over the past years, RGB-D sensors became an important topic in many research areas, including computer graphics, virtual reality, and computer vision. Their popularity increased enormously when Microsoft released its first Microsoft Kinect, an inexpensive, small, and easy-to-use RGB-D camera, which captures rectangular color images and corresponding depth images. Since then, even smaller RGB-D sensors were developed, which are now integrated into many devices. Prominent examples are the Intel RealSense RGB-D cameras, the subsequently released Kinect V2, and Azure Kinect cameras from Microsoft, whose sensors can also be found in the augmented reality headsets HoloLens 1 and 2, and Lidar scanners. Depth images can also be calculated by using two RGB cameras and semi-global matching.

Common applications are, for example, telepresence [20, 55, 326], see also our contribution in Section 3.1, VR/AR applications with remote sensors capturing real world scenes [166], gesture and object recognition and tracking [58], 3D reconstruction [256, 400] and simultaneous localization and mapping (SLAM) [253, 409].

In many cases, the data needs to be streamed over a network beforehand, which runs into the problem of limited network bandwidth, i.e., 100 Mbit/s or 1 Gbit/s for Ethernet, or 300-450 Mbit/s for 802.11n WiFi. The sensors accumulate a large amount of data, and the bandwidth becomes a limiting factor quickly, especially if multiple sensors are combined for better coverage. For instance, a sensor with Full HD color resolution and the depth with a resolution of  $512 \times 424$  (Kinect V2) produces more than 6.6 MB of raw data per frame. At a typical frame rate of 30 Hz, the network would have to support at least 1.60 Gbit/s. More recent depth sensors often support even higher resolutions. For example, the new Azure Kinect is able to capture color, depending on the mode, up to a resolution of 4K, and has a one-megapixel depth sensor.

Data compression is essential in order to reduce the required bandwidth. This enables lower-bandwidth scenarios, makes room for other payloads, and increases the number of possible cameras. For more information about the topic in general, we refer to Section 2.2.5. However, to recap, individual compression of color and depth images is, in general, computationally less expensive than compressing reconstructed 3D data like point clouds or surfaces. Therefore, this approach is often preferred for real-time applications [226]. The color component of RGB-D sensors can easily be compressed with standard image and video compression algorithms like JPEG [391] or H.264 [411] as they are optimized precisely for this task.

However, applying the same encoders to the depth data would often result in sub-optimal compression performance or, more crucially, severe artifacts, and geometric distortions [412]. The reason for this is that depth data usually is represented in a different format, 13 or 16-bit single channel, and has inherently different characteristics than natural color images. In general, depth images consist of more homogeneous regions with abrupt discontinuities at object borders. In addition, individual not-captured pixels and regions of invalid pixels are scattered throughout the image, if not filtered beforehand.

In recent years, the research and development of compression algorithms specially designed for depth images became an important topic. Creating efficient and, at the same time, geometry-preserving (lossless) ones is still a very active area of research.

Our work focuses on analyzing and improving the RVL algorithm [412]. In more detail, our main contributions are:

- An improved adaptive intra-image prediction step for the RVL algorithm, enhancing its compression ratio.
- The addition of a final entropy-coder stage, further enhancing the compression ratio.
- A multi-threaded implementation of the RVL algorithm and variants speeding it up further.
- An additional inter-frame delta step as well as an examination of its effectiveness regarding the compression ratio over different application scenarios.
- Extensive experiments comparing the effectiveness of various lossless depth compression algorithms for different range cameras.

### 3.2.2 Related Work

The output of RGB-D sensors is often visualized in 3D as mesh or point cloud. A lot of focus was put on compressing these geometric representations. In 2007 MPEG issued a call for proposals on point cloud compression to develop an ISO standard [329]. Point cloud compression algorithms are mostly based on spatial data structures like an octree. For example, Mekuria et al. [245] proposed a method that is based on octrees that supports progressive decompression and is also able to exploit temporal redundancies. For the latter, they introduced a predictive coding approach based on macroblocks and rigid transforms between the point clouds of consecutive frames. Thanou et al. [376] introduced a time-varying approach that can predict graph-encoded octree structures between consecutive frames. The idea is to estimate the motion between frames using feature matching based on spectral graph wavelet descriptors and graph-based regularization. Real-time capable mesh-based compression and streaming techniques, like the ones proposed by Mekuria et al. [247] and Banno et al. [15], are in most cases lossy algorithms. As both of these representations, point clouds, and meshes, are three-dimensional, it is more challenging to find and encode redundancy in the data efficiently. Specialized data structures are needed and raise the complexity and computation time for high compression ratios.

A different approach is to encode the raw depth images. Many standard lossless image and video codecs like PNG [303], JPEG-LS [401, 402] or H.264 [411] can be applied, but the results are rather poor, founded in the inherent differences between natural images and depth images [333, 412, 143].

Similarly, general-purpose compression algorithms can be applied to depth images, but intrinsically are not optimized for this kind of data. Therefore they are not ideal solutions.

In recent years some effort was made to adapt common video codes like H.264 and its successor HEVC [363] to suit depth data better. Pece et al. [275] proposed an encoding scheme to convert the single-channel depth data to the RGB format used by H.264 and VP8, reducing the occurring artifacts. It is based on three functions: a linear one for low-resolution information and two triangle-wave functions for the fast-changing fine details. This technique still produces noise at the borders, though. Liu et al. [226] developed a hybrid lossless-lossy algorithm, where the first 2 bits are encoded lossless using run length coding, and the remaining ten bits are compressed using H.264. The HEVC standard [363] features an extension called 3D-HEVC designed for 3D video coding, which uses the multiview texture videos plus depth maps (MVD) format. This extension addresses the compression of depth images through multiple techniques. The complexity is rather high, though. Aimed

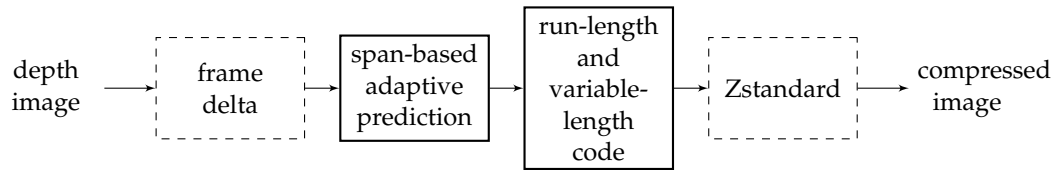


FIGURE 3.13: The pipeline of our approach. Stages in dashed lines are optional. They allow a better balance between compression ratio and compression speed. First stage: frame delta, second: our span-based adaptive intra-image prediction, third: RVL’s run-length and variable-length coding, fourth: an additional Zstandard [74] pass.

at lowering it and enhancing the compression speed, Zhang et al. [438] proposed multiple modifications exploiting depth map and texture video correlation. Further speedups can be gained by limiting costly intra-image compression steps to regions, where they are effective according to a decision model based on tensor feature extraction, as proposed by Hamout and Elyousfi [140].

Recently, Kumar and Ramakrishnan [143] proposed another technique in which even noisy depth images and videos are encoded through planar segmentation. Each frame gets segmented into a number of planes by using the Graph Cut algorithm over the image defined as a Markov Random Field. While the proposed method achieves a high Rate-Distortion performance, it is rather time-consuming and only effectively applicable to scenes that can be approximated by planes.

Most work of depth image compression is about lossy compression. Lossless solutions are quite rare. Mehrotra et al. [243] combined simple intra-image prediction (delta between consecutive pixels in a 1D array), run-length encoding, and variable bit-length coding (Golomb-Rice) for lossless encoding of depth images. Wilson [412] took a similar but even simpler approach and achieves with the proposed RVL algorithm higher speeds at comparable compression ratios. In order to further increase RVL’s compression ratio, Jun and Bailenson [169] adapted the RVL algorithm by applying it to the deltas between consecutive frames, thus, considering the temporal domain, and adding tolerances. With the latter, small changes between consecutive pixel values get ignored, and invalid pixels filled-in with the previous frame’s data. Naturally, this makes the algorithms lossy. Yu et al. [432] instead used simple run-length coding for depth compression, which they found for their specific application to be optimal. Moreover, they employ only 8 bits per pixel while retaining sufficient accuracy, as they limit themselves to a narrow depth range.

### 3.2.3 Proposed Approach

We created a pipeline of four steps (see Fig. 3.13) to compress depth images losslessly from real sensors with a high compression ratio. First, if we have a sequence of images, we optionally calculate deltas between the frames. Afterward, we define spans and calculate individual predictors for each of them, before it will be compressed by RVL’s [412] alternating run-length and variable-length coding. To decrease the lower bound of RVL, we decided to use Zstandard as a final, optional step. The focus lies on captured 16-bit depth values, which can be handled as a single-channel grayscale image. The corresponding color images are not considered in this work.

RVL consists of alternating run-length coding of zero-values, which represents invalid pixels, and variable-length coding for the rest. Like many compression algorithms, RVL compresses not the raw pixel values but the residuals, which remain

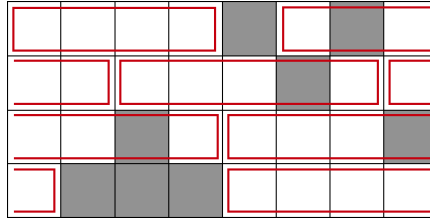


FIGURE 3.14: The pixel grid is segmented in spans of four valid pixels (red boxes). Invalid pixels (grey) are skipped.

after calculating the delta to a prediction of the current value. This leads to a decorrelation of the data, which in turn improves the effectiveness of subsequent compression steps. Smaller residuals have a low entropy, and fewer bits are needed to encode them. In the case of RVL, the prediction of the current pixel value is simply the last valid pixel, which is, in most cases, the pixel to the left.

### Adaptive Span-Based Prediction

One crucial aspect was to improve the simple inter-image delta calculation of RVL to generate smaller residuals. As the employed decorrelation heuristic is not ideal, we propose to replace it with an adaptive selector of different predictor functions. In lossless compression, the transformations and functions applied in the compression stage must be reversible in the corresponding decompression stage; hence, the used prediction method must be encoded in the form of bitflags, too, because all the transformations must be reversible. Pixel-wise switching of the predictors leads to too many bitflags. Therefore the image is dynamically partitioned into spans of valid pixels in our approach. A span can be described as a one-dimensional block of a fixed number of consecutive pixel values. Invalid zero-pixels are skipped. Figure 3.14 shows an example partitioning of pixels into spans of length four.

Our adaptive prediction then works as follows: For each valid pixel  $p$  in the span  $S$ , all predictor functions  $\text{Pred}_i(p)$  are evaluated and the one, which in total leads to the smallest absolute residuals, gets chosen for all pixels in the span to calculate the final residuals  $r_p$ :

$$r_p = \text{Pred}_k(p) \quad (3.1)$$

where

$$k = \underset{i \in [0,3]}{\text{argmin}} \left\{ \sum_{p \in \text{valid}(S)} |\text{Pred}_i(p)| \right\} \quad (3.2)$$

We decided to use four different predictor functions, which then can be represented by exactly two bits per span. In principle, the actual predictors, as well as their quantity, are easily exchangeable, focusing either on computationally simpler and, therefore, faster ones or more complex and effective ones. To predict a pixel  $p$ , we opted to use RVL's standard predictor, given in Equation 3.3, as the default case, as it is similar to the common "left pixel" approach, but handles the occurrences of intermixed zero-pixels very well by skipping them. In Fig. 3.15, this process is illustrated.

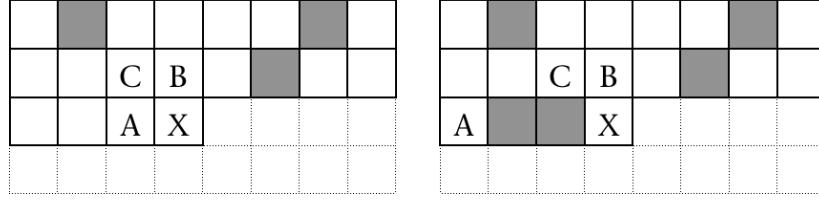


FIGURE 3.15: The prediction for pixel  $p$  depends on the pixel itself ( $X$ ) and its neighbors last ( $A$ ), above ( $B$ ), and above left ( $C$ ). Contrary to  $B$  and  $C$ ,  $A$  is the last valid pixel and, therefore, not fixed, as the right image depicts.

Additionally, we use predictors based on the delta to the upper pixel (Eq. 3.4), the average of the left and upper pixel (Eq. 3.5), and lastly, the result of a combination of the left, upper and upper left pixels (Eq. 3.6).

$$\text{Pred}_0(p) = p_X - p_A \quad (3.3)$$

$$\text{Pred}_1(p) = p_X - p_B \quad (3.4)$$

$$\text{Pred}_2(p) = p_X - \frac{p_A + p_B}{2} \quad (3.5)$$

$$\text{Pred}_3(p) = p_X - (p_A + p_B - p_C) \quad (3.6)$$

Equation 3.6 is the default case in the Paeth [303, pp. 159–162] as well as the MED predictors [401]. We use only this part of them as our fourth predictor because it is computationally less expensive than using the complete Paeth or MED predictor. Tests we conducted showed that using the full MED predictor is much more time-consuming, but does not improve the compression ratio significantly. In this way, the number of additional bits needed for the predictor representation can be significantly reduced, while retaining the ability to adapt to the local image characteristics dynamically. The exact number of predictor bits for the image can be computed as

$$x = \left\lceil \frac{(n - z)}{s} \right\rceil \cdot \lceil \log_2(f) \rceil \quad (3.7)$$

where  $n$  denotes the number of pixels in the image,  $z$  the number of zero-pixels,  $s$  the span size and  $f$  the number of predictor functions.

### Inter-Frame Delta Computation

Another aspect we concentrated on is adding a frame delta component as a first step in the algorithm's pipeline. This is a commonly used technique in video compression, where differentials between subsequent images are encoded instead of individual images one by one. Figure 3.16 illustrates the process. The effectiveness depends on the application scenario, the content, and how dynamic it is. At least in cases where the change between the images is small, or only some dynamic elements occur, this technique should be beneficial for the compression ratio and speed. In the original RVL paper [412], it is mentioned that such a frame delta calculation was experimented with but the compression was even worse. According to the paper, the test scene was a dynamic scene in which the camera constantly moved, which

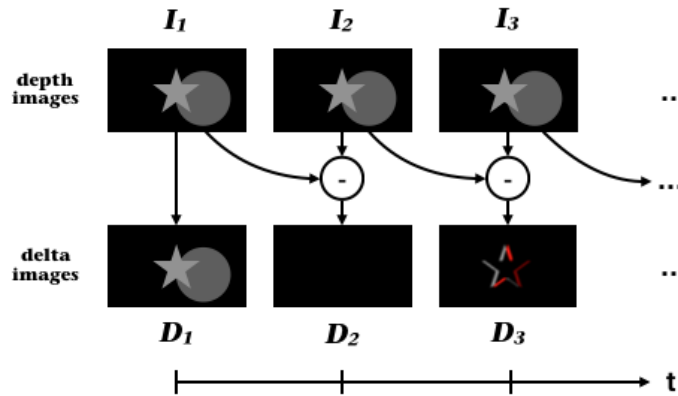


FIGURE 3.16: Frame delta computation: Sequence of images  $I_i$  and the corresponding differential images  $D_i$  between every two consecutive ones.

would be the worst-case scenario for this kind of technique. It is not clear if other scenarios were tested. For our pipeline, we designed the frame delta computation as an optional first stage so that it can be skipped in scenarios where it is not effective. Another aspect to consider is the inherent noise of the sensor. Some pixels switch between being valid and invalid, and in addition, the depth readings continuously vary, even for physically static objects. This results in a decreased effectiveness of frame delta computation but can be compensated for by temporal filtering as a pre-processing step (at least to some degree). Depending on how intelligent and vigorous the employed filter is, other artifacts may be introduced, which could be tolerated depending on the application.

### Further bit reduction

The coding of the residuals in RVL is done by variable bit length, where each valid pixel is at least one nibble (four bits) long. For each nibble, the first bit functions as a continuation bit, and the other three bits are used to represent (a part of) the actual value of the residual. Therefore, in a single nibble, a residual  $r \in [0, 7]$  can be represented.

In cases where image regions are very homogeneous, and the computed residuals are frequently below this maximum value of seven, the algorithm can lead to comparatively poor compression results. We propose to couple RVL with a second encoder without such a limitation to mitigate this drawback. In our approach, the compressed output of RVL is, therefore, further processed by Zstandard [74]. As a combination of a dynamic dictionary-based component with a sliding search window and an ANS-based entropy component, Zstandard can potentially compress symbols smaller than four bits. According to empirical tests, Zstandard performed best as a backend.

### Parallel Execution

To mitigate the inevitably increasing computation time by the more complex prediction, we implemented multi-threaded versions. The principle is the same for all variants: first, for each thread, all necessary data and an output buffer are initialized,

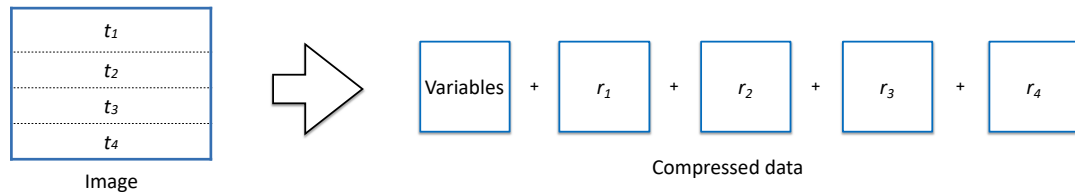


FIGURE 3.17: Multi-threaded compression of the depth images. Each image is partitioned into equally-sized blocks which then get processed simultaneously by  $t_i$  threads. The resulting compressed parts  $r_i$  eventually get concatenated together with a header-block.

then the image, represented as a one-dimensional buffer, is segmented into blocks by the number of threads. Lastly, the compressed parts are stitched together into one continuous block, see Fig. 3.17. Especially our improved prediction step in the compression stage should benefit from parallel execution as it is computationally the most expensive part and there are only locally very confined dependencies between the pixel computations.

### 3.2.4 Results

All of our tests were conducted on an Intel Core i7-7800X, 16 GB of system memory, and under Windows 10 x64 Enterprise using the Visual Studio 2017 compiler. Each test was performed multiple times, and the average was taken.

We recorded sequences of depth images of six different test scenes with the Azure Kinect RGB-D camera in both of the two available modes, narrow and wide field of view (NFOV, WFOV). In NFOV, the depth was recorded in  $640 \times 576$  at 30 Hz and in WFOV at  $1024 \times 1024$  at 15 Hz. Each depth value is represented as a 16-bit integer (short). Four of those scenes were static, and two were dynamic. In the first dynamic scene, the camera was fixed, and a person moved in front of the camera. In the second scene, the camera was handheld and moved around. For the static scenes, 30 frames were recorded, while for the dynamic ones, 120 frames were used. Additionally, we tested three single depth images from the Middlebury dataset [320] with very homogeneous depth values, and a dynamic scene of 600 frames from TUM [360], in which the camera (Kinect 1) was handheld, slightly shaken, and people moved, while seated.

In each test case, we omitted the first recorded frame from the evaluation as we did a lot of initialization work for the algorithms here (e.g., reserving memory) which holds for the rest of the test. The measurements of all the other subsequent frames in a test were then averaged.

The Azure Kinect camera has the unique attribute that the output depth-image is rectangular, but depending on the mode, the content is only hexagonal or spherical. The corners are zero-pixels. While this is the standard output of the camera and could be representative of other cases, where significant static areas fail to get captured by an RGB-D sensor, it is a rather uncommon situation. In order to not only test the standard configuration but also more broadly comparable scenarios, we also evaluated depth maps cropped by 25%. As a result, most of the static invalid regions were dropped.

Figure 3.18 shows a selection of our test scenes. For an overview of all scenes and corresponding metrics, we refer to the Figures 3.24 and 3.25 as well as Table 3.3



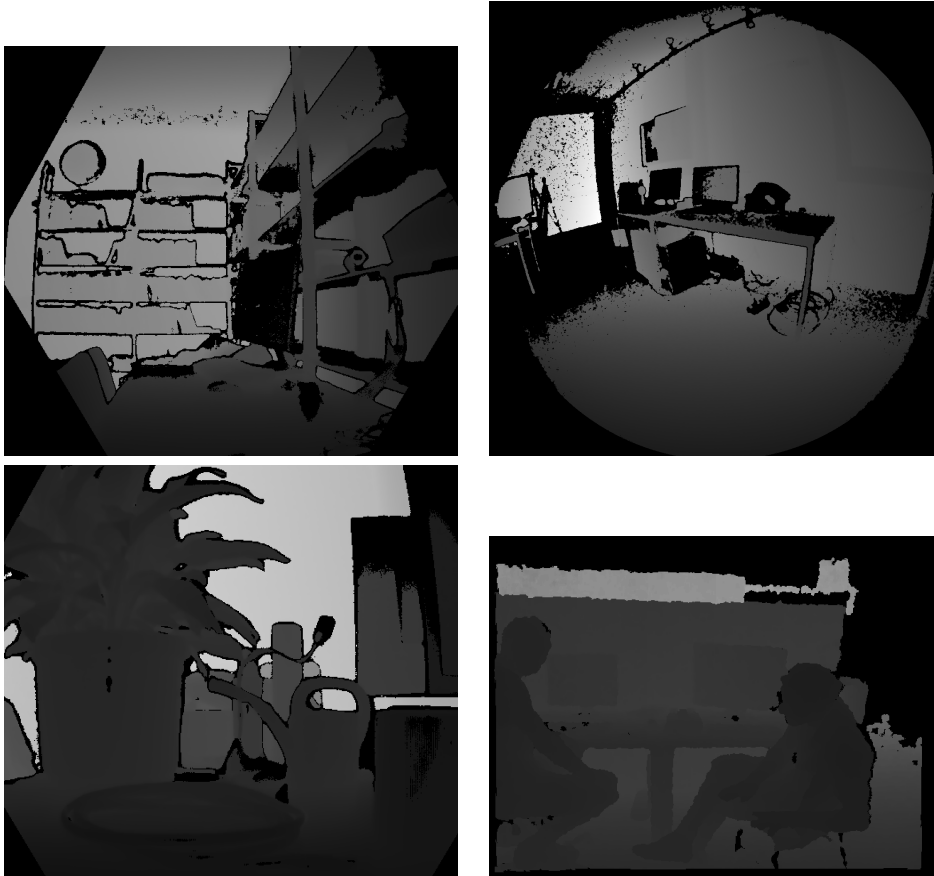


FIGURE 3.18: Example depth images of four different scenes. We recorded the first three with the Azure Kinect in different modes (NFOV, WFOV, NFOV with cropping). The last scene from TUM [360].

at the end of this section.

We compare our novel compression algorithm against the original RVL algorithm [412], PNG [303] as a classical lossless image compression algorithm that is widely used, and LZ4 as a fast representative for the LZ77 family of dictionary-based encoders. Additionally, as an example of a modern and efficient entropy coder, we decided to use an ANS implementation by F. Giesen [128]. In empiric tests, this implementation performed best. Lastly, Zstandard as a dictionary coder is also compared.

For our algorithm Pred, we chose a span size of 16 pixels, and for the PredZ variant eight, as well as a value of two for the Zstandard, pass. These configurations yielded the highest compression ratios, while still achieving interactive speeds. For algorithms featuring a selectable acceleration (or compression) factor, we tested them in multiple configurations and chose the one, which minimized the difference of the compression ratio in comparison to our algorithm. In the case of PNG, a quality level of five was chosen, for LZ4, an acceleration factor of one, and for Zstandard a value of six.

The first test we conducted is a general comparison of all the algorithms in terms

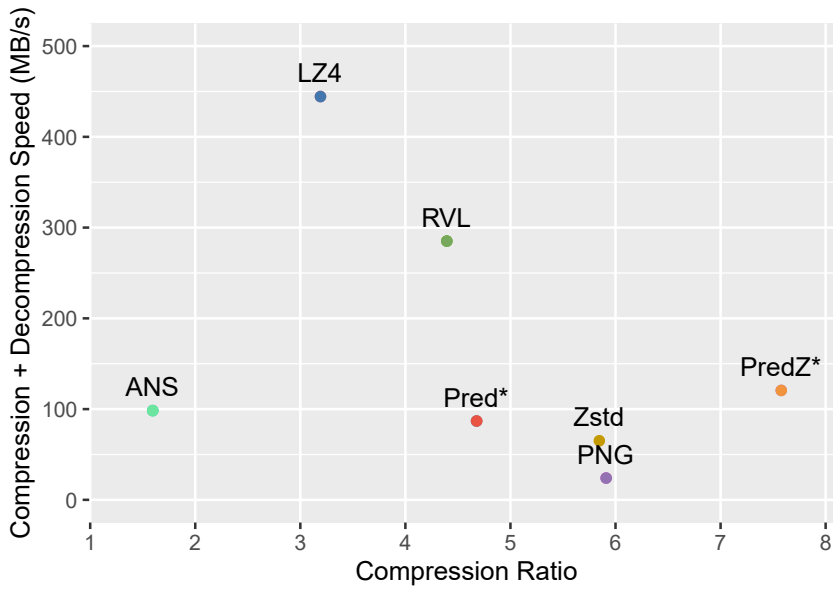


FIGURE 3.19: Average combined compression and decompression speed over compression ratio for all algorithms. Algorithms with asterisk mark our algorithms. Our PredZ algorithm achieves the highest absolute compression ratio, while still delivering real-time performance.

of their respective compression ratio  $cr$  and combined compression and decompression speed  $sp$ . Equations 3.8 and 3.9 describe the calculation of the metrics.

$$sp = \frac{2 \times ob}{ct + dt} \quad (3.8)$$

$$cr = \frac{ob}{cb} \quad (3.9)$$

$ob$  stands for the original image size in megabytes,  $ct$  and  $dt$  for the compression and decompression times in seconds, and  $cb$  for the compressed size in megabytes.

Figure 3.19 presents the results of this initial comparison, where for each algorithm, the thread configuration was selected, which yielded the best compression rate; therefore, the number of threads was not equal for all algorithms. For this test, the mean over all scenes and modes were taken. The asterisks identify our algorithm variants. It can be seen that, on average, ANS is not competitive in our test, neither in speed nor compression ratio. LZ4 is the fastest but compresses rather poorly. PNG has a high compression ratio of 5.9:1 but at the cost of a very slow combined speed. RVL achieves good performance and a compression ratio of nearly 4.5:1. Our span-based adaptive prediction Pred can boost the compression ratio on average by 6.5 %, but the performance decreases considerably. A possible explanation for this drop in performance, despite the rather low complexity predictors, might be the lack of optimization compared to RVL. Interestingly, the general-purpose Zstandard algorithm can achieve an even higher compression ratio of roughly 5.8:1. However, it is also slower. Lastly, with our most sophisticated variant PredZ, we achieve the best compression ratio of more than 7.5:1, which is significantly higher than the ones

TABLE 3.2: Distribution of adaptively selected predictors, averaged over all scenes and modes.

Predictor ID	0	1	2	3
Usage %	24.4	26.6	21.2	27.7

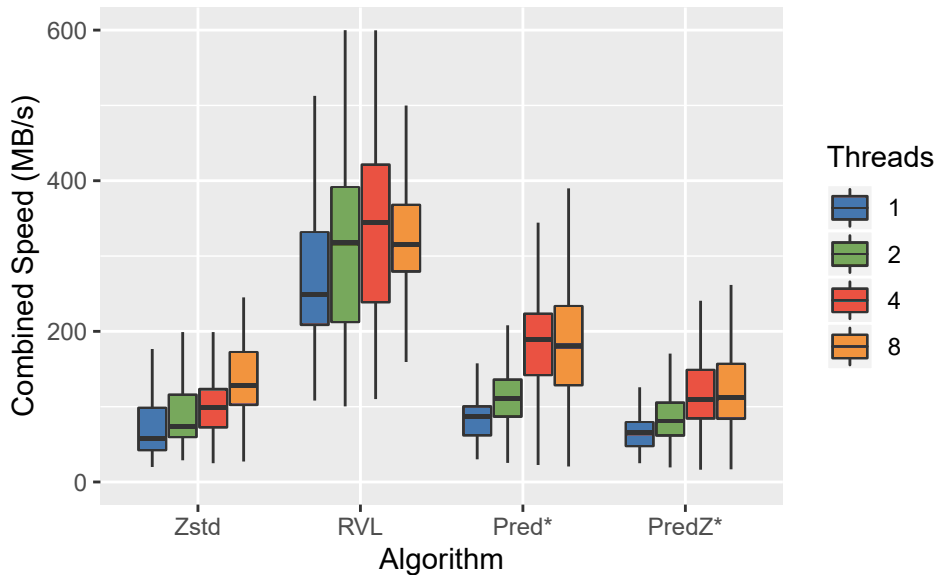


FIGURE 3.20: The image illustrates the impact of parallelization on the combined compression and decompression speed for different algorithms. All algorithms including RVL and our RVL-based ones benefit from parallel execution.

of RVL or Zstandard. At the same time, the combined compression and decompression speed of 121.8 MB/s is more than sufficient to maintain interactive frame rates. For more comprehensive data of this test, we refer to Figure 3.26 at the end of this section.

To further analyze the effectiveness of our span-based adaptive prediction component, we counted the frequency of how often each predictor is chosen as the best one per image. Table 3.2 shows the percentages averaged over all scenes and frames.

Each predictor gets equally used, which indicates that the adaptive prediction works as intended, and all of the chosen predictors complement each other to effectively reduce the average residual in contrast to a static prediction like it is used, for example, in the original RVL algorithm.

For all subsequent tests, we only consider the most promising algorithms and omit the ones without competitive results, namely ANS, LZ4, and PNG.

In order to review our multi-threading implementation and analyze the behavior of the algorithms with increasing parallel execution, we performed all tests as well with two, four, and eight threads. Measuring initial thread creation overhead is prevented by early thread allocation. The multi-threading performance is shown in Fig. 3.20, in general, the combined speed rises.

As expected, the performance gains shrink with more threads as a result of diminishing returns. However, we were able to achieve considerable speedups for

RVL, although it has low arithmetic complexity and, thus, becomes quickly memory-bound. With four threads, the performance increased by a factor of 1.42. Our more complex RVL-based algorithm benefits highly from the parallel execution, as increasing the threads from one to four leads to a speedup of 2.16. While all algorithms gain performance by multi-threading, at least for up to four threads, the speedup is generally much lower than the theoretical optimum. This may be partly because not all parts of the algorithms are multi-threaded, e.g., merging the results, and because some sections of the algorithms are rather bandwidth- than compute-bound. An evaluation of the timings of the individual parts of our Pred algorithm indicates that the more computationally complex prediction stage benefits significantly more from the parallelization than the run- and variable-length coding stage. Furthermore, the decompression component does not profit as much as the compression component. The compression ratio did slightly, but not considerably, decrease with more threads, as was expected due to the separated image blocks. In the case of Zstandard, the compression ratio decreased the most with 2.68 %. RVL, on the other hand, had a decrease of less than 0.1 %. The compression ratio of our adaptive prediction dropped by 0.25 %.

To analyze the effectiveness of frame delta coding in RVL, each test is executed with and without our implementation of the addition as well as with and without a preceding temporal filter (see the last paragraph of Section 3.2.3), which makes four variants. The filter we implemented and used is rather simple and only meant as an example. Nonetheless, we aimed at preserving the legit information (in contrast to the sensor noise) and avoiding motion artifacts. It works as follows: Pixels of an incoming image will get ignored, whose delta to the corresponding pixels of each of the last two images is below 2 %.

The combination of frame delta coding and temporal filtering increases the compression ratio for all algorithms the most, as is shown in Figure 3.21. While the combination of frame delta coding and temporal filtering, in general, leads to substantial improvements, our algorithm PredZ still achieves the highest compression ratio of up to 13.8:1. It should be noted though, that with temporal filtering, the compression pipeline is not strictly lossless anymore.

With only the frame delta extension, the compression ratio still increases for Pred and RVL, which contradicts the statement of [412]. It should be considered that for our tests we took the average of static and also highly dynamic scenes. Nonetheless, RVL's compression rate increased by 18.54 % and with our improved prediction by 11.1 %. Zstandard does not benefit from frame delta, but only from the temporal filter instead, although the confidence interval is very wide. Surprisingly, the data indicates that our PredZ performs best with the raw data compared to the others.

A detailed breakdown w.r.t. different classes of scenes and their influence on the compression ratio, while computing frame delta, can be seen in Fig. 3.22.

The results indicate that the type of the scene, static or dynamic, or more specifically, the amount and the intensity of movement of the content and the camera itself, do have a significant impact on the compression rate. Without temporal filtering, the average compression rate of all algorithms is about equal. According to Fig. 3.21, not all algorithms benefit directly from frame deltas. If, however, such a filter is used, the influence of the scene type raises strongly, specifically rather static scenes can be compressed extremely well. Interestingly, sometimes the scene which involves camera movement is better compressible using both, the temporal filter and frame delta, instead of the scene, where the camera is static and only parts of the captured environment move.

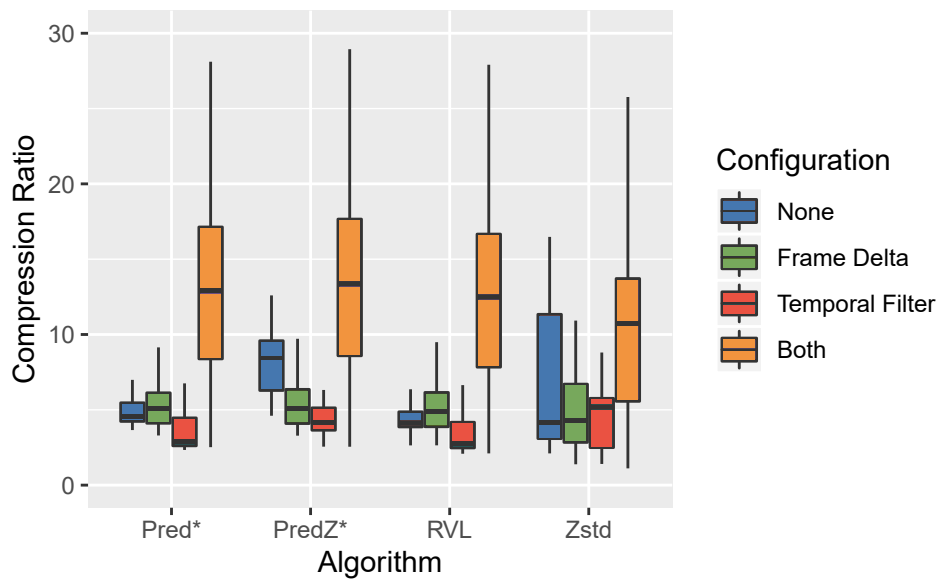


FIGURE 3.21: Comparison of the impact of our inter-frame delta computation on the compression ratio, both with and without a preceding temporal filter. The compression ratio increases hugely with the combination of temporal filter and frame delta, but using frame delta computation without the preceding filter also, most often, accomplishes notable gains.

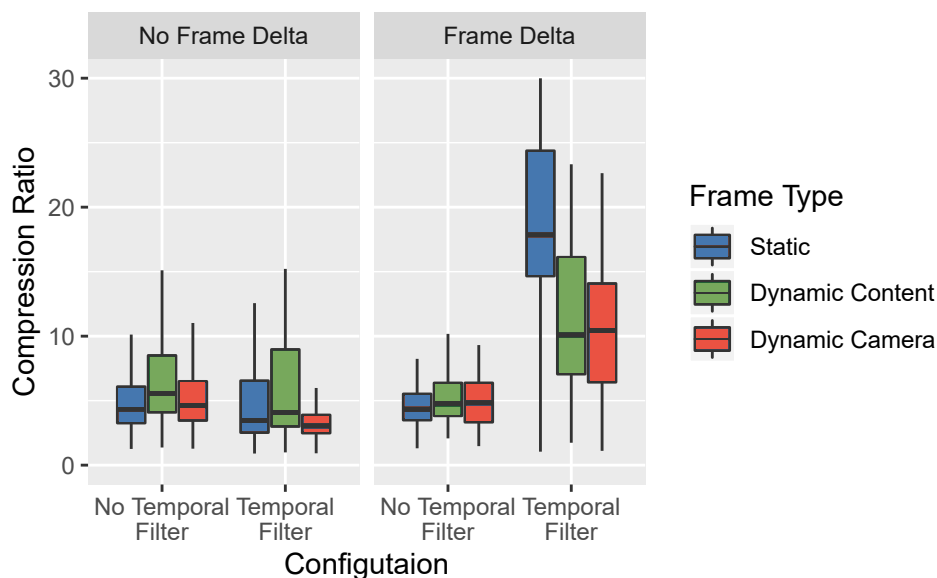


FIGURE 3.22: The image illustrates the impact of the scene type (static, dynamic content, dynamic camera) on the performance of the frame delta computation with and without filtering. Static scenes benefit the most from the combination of temporal filtering and frame delta computation.

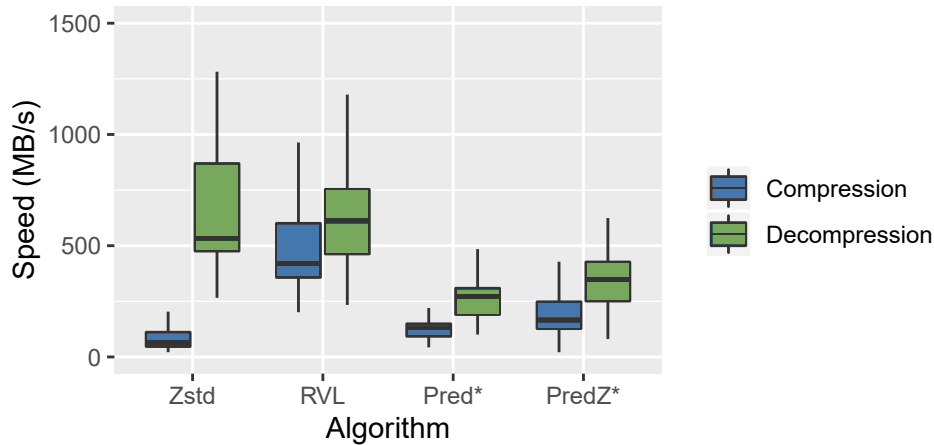


FIGURE 3.23: The image displays the individual compression- and decompression speed for different algorithms. The decompression is significantly faster for everyone.

TABLE 3.3: The table contains the number of pixels and the percentage of invalid zero-pixels for each test scene and mode.

Narrow Mode			Wide Mode			Narrow Cropped Mode			Wide Cropped Mode		
Scene	# Pixels	% Zeros	Scene	# Pixels	% Zeros	Scene	# Pixels	% Zeros	Scene	# Pixels	% Zeros
0N	368,640	33.9	0W	1,048,576	41.6	0NC	207,360	25.6	0WC	589,824	25.7
1N	368,640	22.5	1W	1,048,576	59.0	1NC	207,360	8.6	1WC	589,824	44.5
2N	368,640	26.4	2W	1,048,576	50.0	2NC	207,360	12.6	2WC	589,824	25.7
3N	368,640	42.0	3W	1,048,576	66.6	3NC	207,360	28.3	3WC	589,824	58.9
4N	368,640	28.4	4W	1,048,576	62.5	4NC	207,360	15.6	4WC	589,824	46.7
5N	368,640	30.0	5W	1,048,576	38.6	5NC	207,360	18.1	5WC	589,824	21.7
6	1,542,900	1.0									
7	1,542,900	0.4									
8	1,542,900	1.0									
9	307,200	32.0									

How the combined speed of the tested algorithms is distributed over compression and decompression can be seen in Fig. 3.23. The decompression speed generally outperforms the compression speed, especially in the case of Zstandard.

### Additional Data

Here, we want to show additional data and test results. First, Fig. 3.24 and Fig. 3.25 depict an overview of all the scenes and modes that the test data set consisted of. Then, Table 3.3 lists statistics regarding the number and ratio of invalid pixels for each test scene. Lastly, Fig. 3.26 shows a comparison of the compression algorithms regarding speed and compression ratio in which all data samples (all scenes, modes, frames) are included.

### Evaluation of Lossy Video Compression

In conjunction with the evaluation of our proposed algorithm and other lossless compression algorithms, we did also our own tests regarding the performance of typical lossy video compression algorithms. We did this in order to get a more comprehensive overview of the field and all possible options, get a better understanding

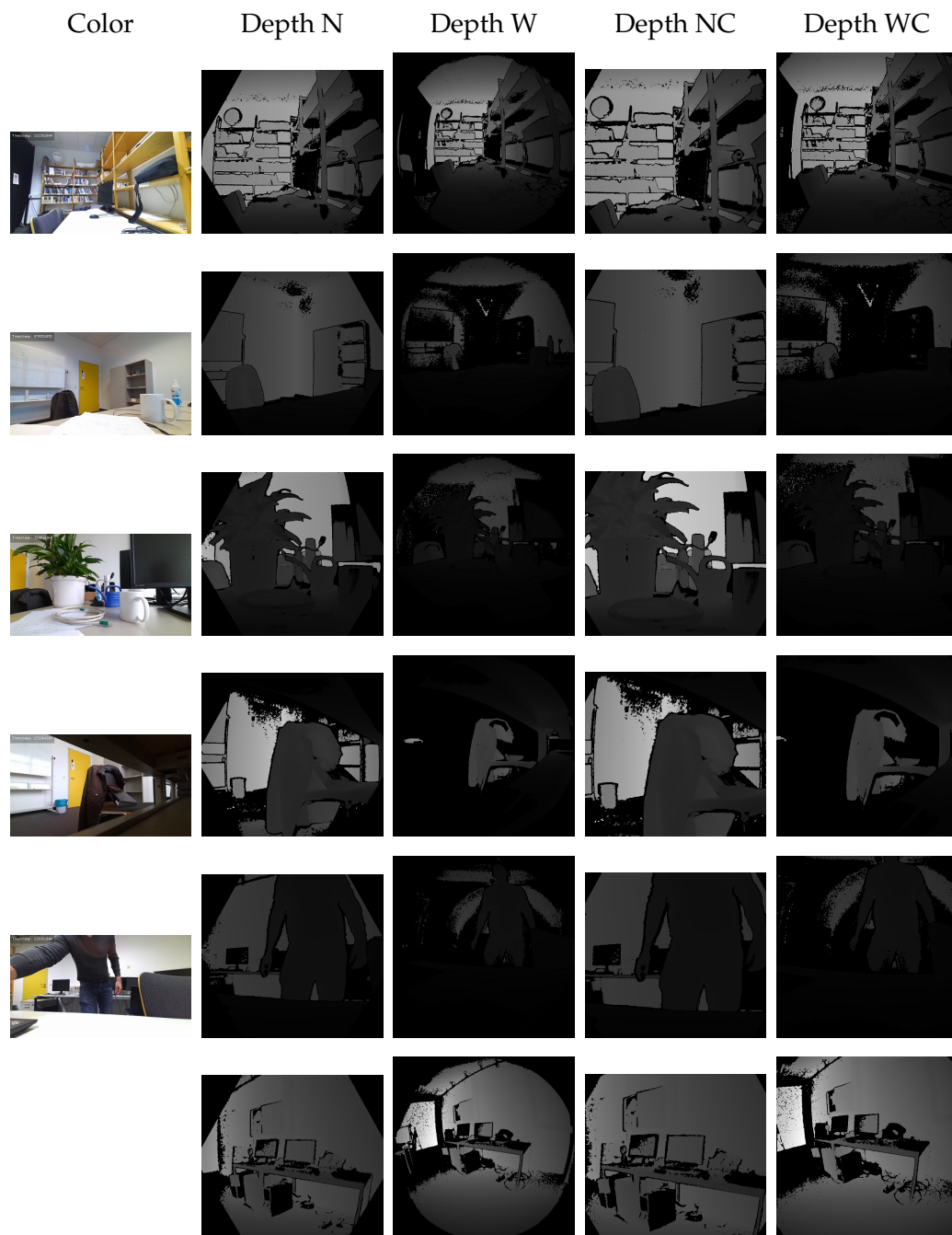


FIGURE 3.24: A table of the test scenes 0 to 5 (which are the self-recorded ones; one scene per row), including the color image, the narrow and wide field of view depth images (Depth N and Depth W), as well as the corresponding cropped versions (Depth NC and Depth WC).



FIGURE 3.25: A table showing the rest of the test scenes (scenes 6 to 9; one per row), consisting of the color and depth images.

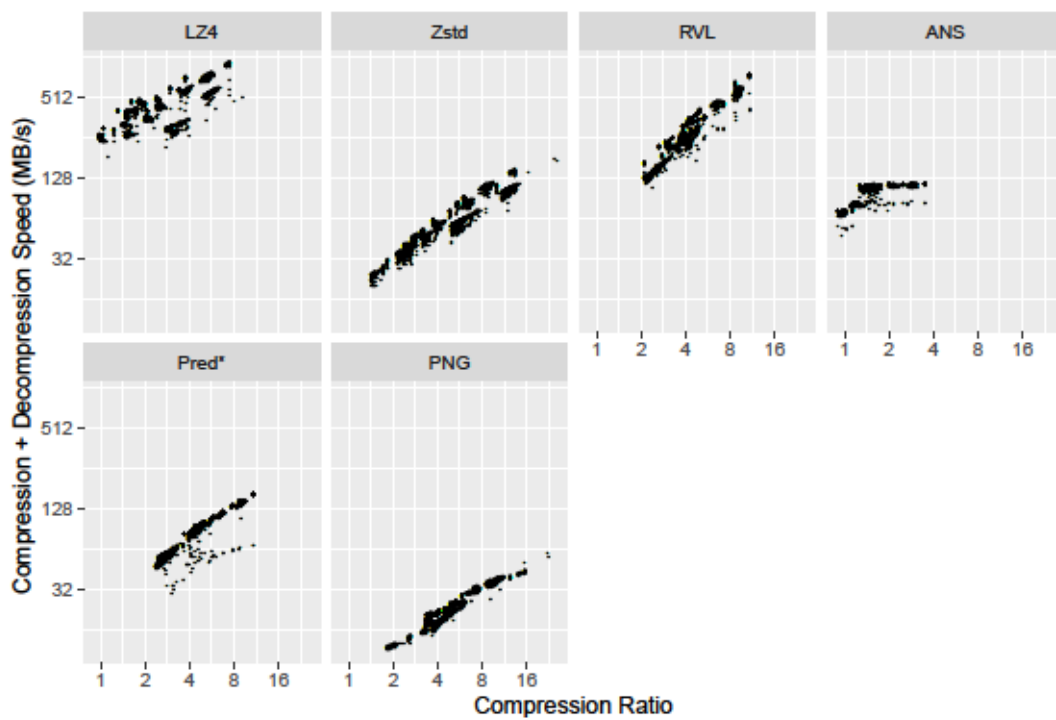


FIGURE 3.26: Overview of compression ratio and combined speed for each algorithm. Each dot represents one measured frame of any scene or mode.



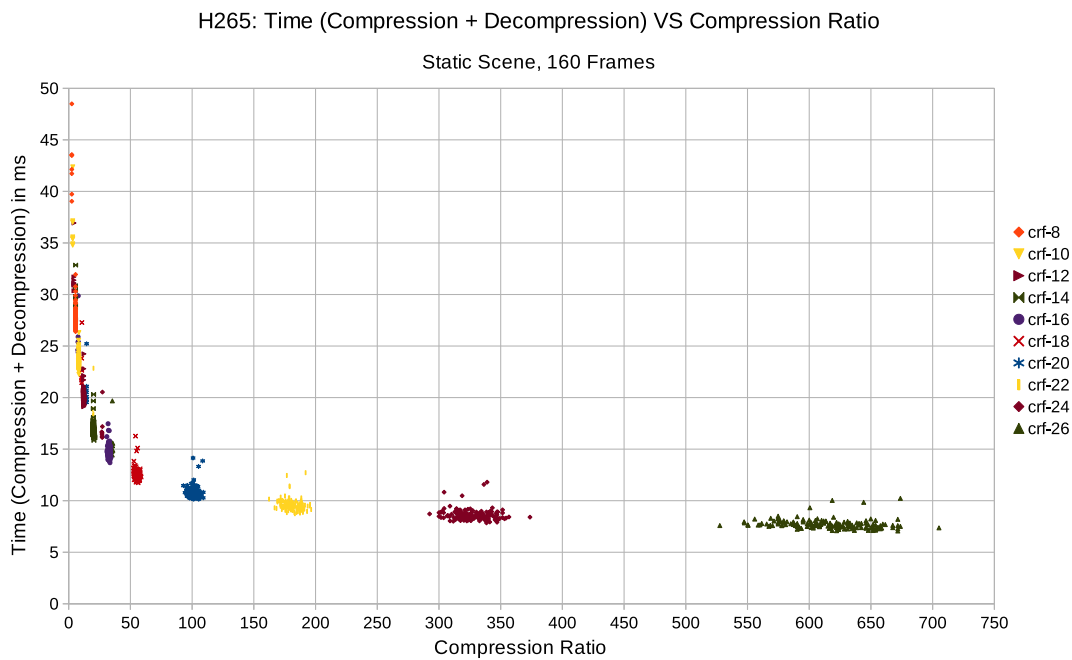


FIGURE 3.27: Evaluation of the video compression algorithm H.265 on depth images regarding time and size over different CRF values. The test scene was static.

of our algorithm's performance in relation to them, and investigate their potential for depth image compression. We chose to test H.264, H.265, AV1, and VP9, as these are common and state-of-the-art video compression algorithms. After initial tests, we already discarded AV1 from further testing, as it was too computationally expensive and, thus, took too much time to be real-time capable. For the other algorithms, we extensively investigated the provided compression parameters (e.g., CRF) in order to get a feeling for the algorithms' behavior and to find a good trade-off between compression ratio and speed. Figure 3.27 compares the compression ratio with the time needed for compression and decompression, in dependence on the CRF value, using H.265 as an example. In this case, a static scene was used. Looking at the plot, the algorithm is very flexible with compression ratios ranging roughly between 5 and 650, and the corresponding time between roughly 30 ms and 7 ms. As expected, the compression ratio decreases while the needed time rises, when lowering the CRF value. Naturally, the lower the CRF value, the higher the resulting image's quality and the more accurate it is regarding the original image. How accurately the images are after compression can be seen in Figure 3.28, which compares the peak signal-to-noise ratio (PSNR) value with the compression ratio, in dependence of the CRF value, again using H.265 and a static scene. The PSNR values range roughly between 44.2 and 41.1. Interestingly, we can see a second curve of clusters. The likely reason for this is that not all frames are processed equally: On most frames, inter-frame compression is applied, meaning, the difference between frames is compressed instead of the original frame itself. However, depending on the parameters (e.g., group of pictures (GOP) size), each  $n$ th frame is again compressed individually, which is less efficient. Especially in this case with a static scene, the difference between using inter-frame compression or not is more pronounced.

We did the same evaluations using a dynamic scene, the results regarding time

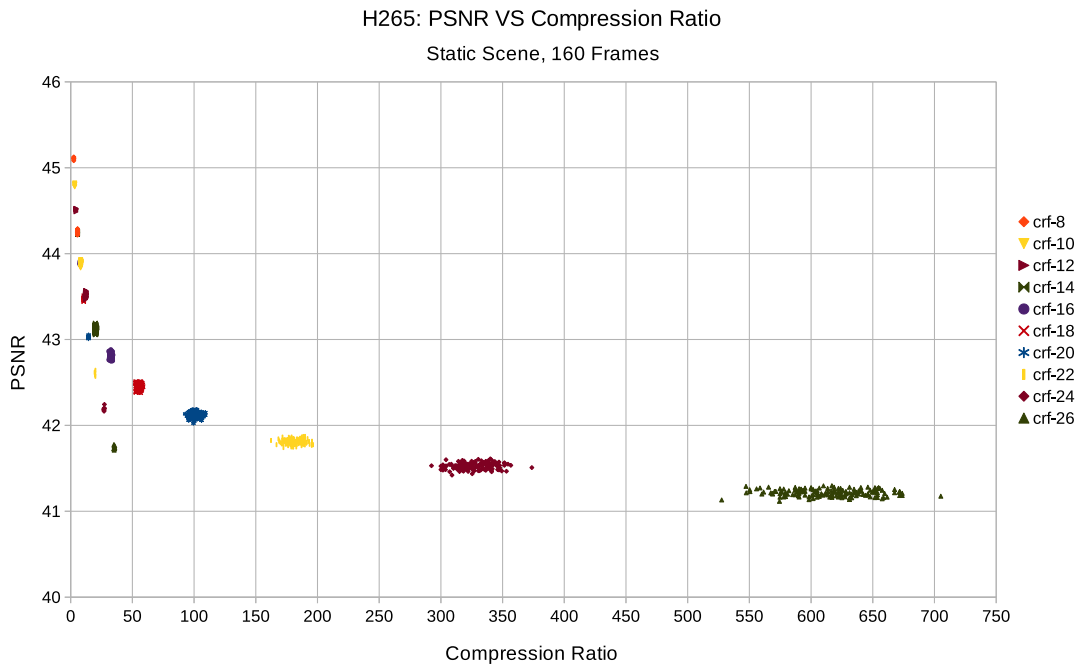


FIGURE 3.28: Evaluation of the video compression algorithm H.265 on depth images regarding PSNR and size over different CRF values. The test scene was static.

and PSNR relative to the compression ratio and CRF value are depicted in Figures 3.29 and 3.30, respectively. As we can see, in comparison to the static scene, the individual clusters are less homogeneous and more spread out, which is to be expected with the more varied frames from the dynamic scenes. Also, the effectiveness of inter-frame compression is reduced, thus, we do not see as clear the second curve of clusters as in the static example and, generally, lower compression ratios.

Eventually, we did a direct comparison of the three video compression algorithms regarding PSNR, size, and time. We chose parameters that yielded a good trade-off between speed and compression ratio. For instance, we chose a CRF value of 20, the “ultrafast” preset, and the “zero-latency” setting. Table 3.4 shows the results (in terms of medians) of the comparison. H.264 is the fastest and VP9 is by far the slowest. H.265 is in between and still very quick. Moreover, H.264 is also the most effective in compressing the images (at this CRF value) but at the cost of the worst PSNR values. H.265 has just slightly better compression ratios than VP9. However, VP9 achieves the highest PSNR values. Figure 3.31 depicts the algorithms’ performance in the form of a time series over one test scene regarding the compression ratio, Figure 3.32 regarding the needed time, and Figure 3.33 regarding the resulting PSNR. Note the logarithmic scale on the y-axis of Fig. 3.32. We can observe, that H.264 performs best regarding time and compression ratio and H.265 follows suit at a moderate distance. VP9 is just slightly worse than H.265 in terms of compression ratio but also has many outlier frames that compress way worse. VP9’s compression time is way worse than the others, though. However, looking at the PSNR values, the behavior flips and VP9 gets the best results and H.264 the worst. We can also see that VP9 and H.265 have periodically occurring spikes, which likely result from these frames being the start of the GOP for the inter-frame compression. Considering all three factors, speed, PSNR, and compression ratio, H.265 achieves

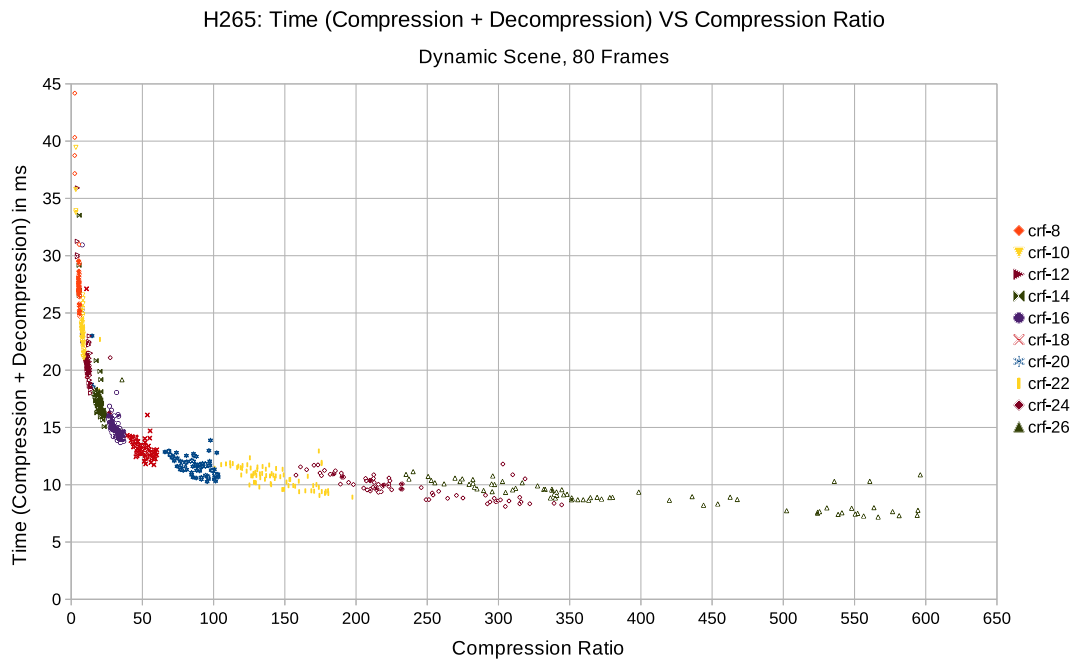


FIGURE 3.29: Evaluation of the video compression algorithm H.265 on depth images regarding time and size over different CRF values. The test scene was dynamic.

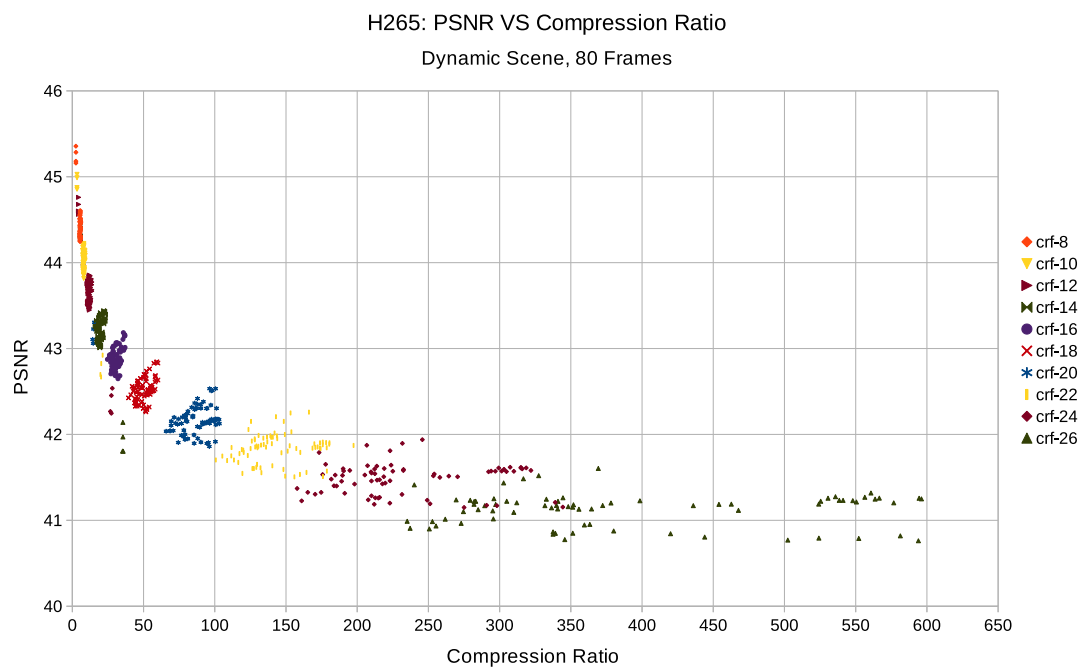


FIGURE 3.30: Evaluation of the video compression algorithm H.265 on depth images regarding PSNR and size over different CRF values. The test scene was dynamic.

TABLE 3.4: Comparison (median) of the video compression algorithms H.264, H.265, and VP9 regarding time (compression + de-compression), compressed size, and PSNR. The best values are highlighted in bold.

Metric (Mdn.)	H.264	H.265	VP9
Time (ms)	<b>1.88</b>	2.47	223.00
Size (KB)	<b>0.71</b>	1.23	1.24
PSNR	40.56	41.30	<b>41.56</b>

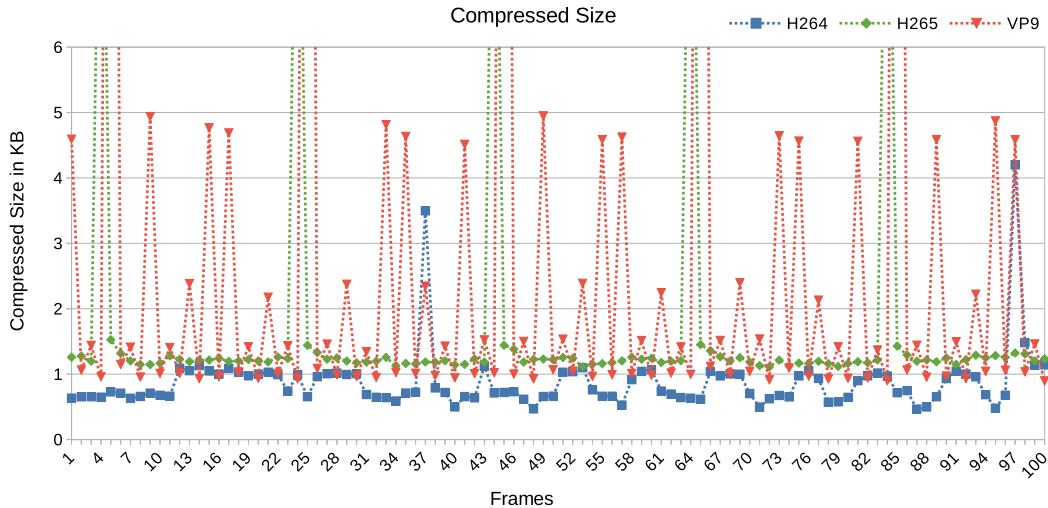


FIGURE 3.31: Comparison of the video compression algorithms H.264, H.265 and VP9 regarding the compressed size of a test scene. H.264 performs the best and VP9 the worst, although just slightly.

the best and most balanced results. Naturally, each algorithm and its performance can be tweaked to some degree to the specific requirements.

Generally, even though the measured compression speeds, compression ratios, and PSNR results look promising, all these video compression algorithms produce visible, severe artifacts, as previous work already showed. See for example Figure 3.34. We can therefore reinforce the notion that simply applying these algorithms to depth data is not a viable option. However, we also see the great potential for lossy and near-lossless compression, which possibly can be exploited if the algorithms could be adapted to better suit depth data, even though this is not trivial, as previous work also discovered.

### 3.2.5 Conclusions and Future Work

We proposed a lossless RVL-based algorithm for real-time depth image compression aimed at high compression ratios. Our experiments show that our algorithm achieves the highest compression ratios compared to competing real-time capable algorithms. With our method, depth images can be compressed up to 73 % smaller than with the original RVL algorithm and 30 % smaller than with the slower Zstandard. Using temporal filtering beforehand, we accomplish even higher compression ratios of 13.8:1, which is still the highest compared to the other evaluated algorithms but the compression pipeline becomes lossy. Thanks to parallel execution,

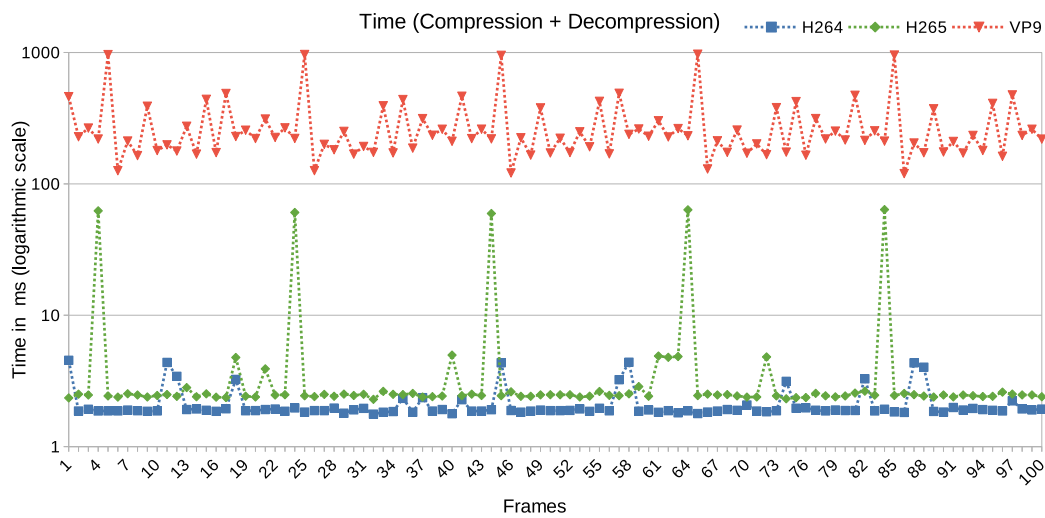


FIGURE 3.32: Comparison of the video compression algorithms H.264, H.265 and VP9 regarding the time (compression + decompression) of a test scene. Note the logarithmic scale on the y-axis. H.264 performs the best and VP9 by far the worst.

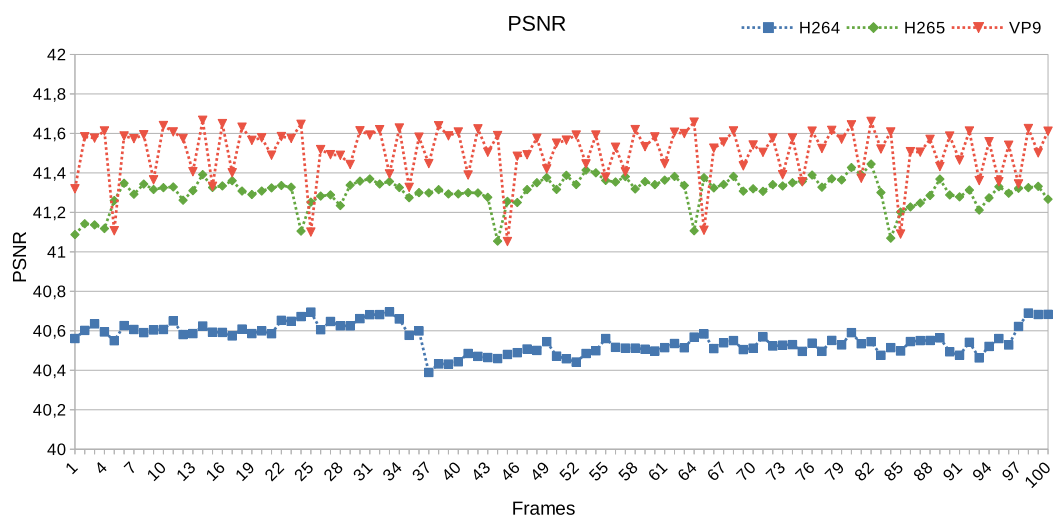


FIGURE 3.33: Comparison of the video compression algorithms H.264, H.265 and VP9 regarding the PSNR value. VP9 performs the best and H.264 the worst.

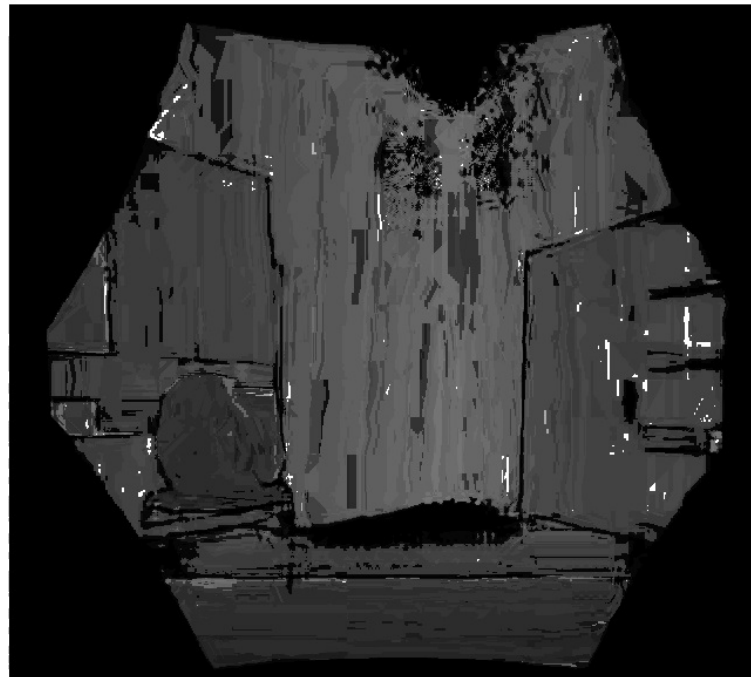


FIGURE 3.34: Artifacts (white areas, grey block/stripe patterns) produced by standard video compression algorithms (e.g., H.265) when on depth images. Note, the image's brightness was increased non-linearly for visualization purposes.

our performance is still sufficient for typical RGB-D-sensor frame rates and real-time streaming in bandwidth-limited applications. Finally, we were able to show that the original RVL can also be sped up with multi-threading, and it can, for some use cases, benefit significantly from frame delta computation. As future work, further improvements could be made by optimizing the span-based prediction stage to lower the computation times, and maybe even SIMD can be applied. Also, it may be worth investigating zigzag scanning and using a block-based adaptive prediction instead. Moreover, an accompanying evaluation of lossy video algorithm algorithm showed that they hold potential (e.g. for near-lossless compression) but require further research and modifications to prevent artifacts.

### 3.3 Enhancement of Depth Images using Deep Neural Networks

As briefly described at the start of the chapter, not only reducing the size of the depth images using compression is important to achieve distributed high-quality live 3D reconstructions of the captured RGB-D data, but also enhancing the imperfect depth images themselves. Specifically, reconstructing missing areas in the depth images using inpainting techniques. In recent years, the advantages of deep-learning-based color image inpainting became prominent. However, research on the inpainting of depth images as captured by RGB-D cameras is scarce.

In this section, we, therefore, describe our approach of utilizing existing deep learning models to investigate the reconstruction quality of missing areas in depth images. Concretely, we chose a U-Net architecture with partial convolution layers

and a generative adversarial network architecture that takes advantage of a patch-based discriminator. For comparison, we took a basic U-Net and the state-of-the-art model called LaMa. The training was done on datasets augmented with synthetically generated noise/holes and using an elaborate preprocessing pipeline. Our results show (at least) reasonable good and coherent results for all models using three different datasets. Moreover, all models except for LaMa achieved real-time inference speed.

The work presented in this section is based on our submitted paper PP2 in Appendix A.

### 3.3.1 Introduction

We already discussed, that with the growing availability of low-cost depth sensors and RGB-D cameras, such as the Azure Kinect, their popularity and employment increased throughout various research areas and industries. To recap, typical use cases are, for instance, SLAM and object detection in computer vision and robotic applications, or real-time capturing of point cloud avatars for telepresence systems. A long-lasting challenge is, however, handling the inherent sensor noise, as well as artifacts and holes that lead to an incomplete depth image. As explained in Section 2.2.3, these issues are inevitable consequences of the TOF principle many depth sensors use. Concretely, multipath inference, caused by repeated reflection of the infrared rays between objects, and signals that are too powerful or too weak lead to ambiguous or invalid depth values. Having accurate and dense depth maps is important for many downstream tasks such as safe motion planning, reliable vision in autonomous vehicles, remote surgery assistance, or industrial inspection. One solution would be to adapt those downstream, domain-specific algorithms and models to work with incomplete input. However, experience shows that specialized algorithms and models that only focus on the specific task of denoising/inpainting the input data are more effective and lead to better overall results.

Therefore, preprocessing and enhancing the depth images is an important task. Reconstructing the missing areas in real-time is not trivial, though, as there are strong spatial dependencies between the data points, both locally and globally. Additionally, previous work in the area of hole filling and image inpainting was mostly focused on regular color images and is not necessarily well-suited for direct application on depth images. Although we already considered the issue of depth image enhancement while developing the telepresence system that we proposed in Section 3.1, it was not the main focus then and leaves much room for improvement.

With this work, we propose an approach of real-time depth image inpainting using neural networks. Our main contribution is the investigation of the depth image reconstruction quality of two fast U-Net-based network models that were originally designed for color image inpainting, including a comparison with a basic U-Net and a more sophisticated state-of-the-art model. In contrast to many others, the models we use do not need any color images for guidance, which makes them more generally applicable as they can be also employed in use cases where no color information is available. The first model we chose uses partial convolutions, while the second one is based on a GAN architecture. Furthermore, we present a detailed quantitative and qualitative evaluation using two public datasets and a custom one we recorded ourselves.

### 3.3.2 Related Work

Traditionally, missing areas in pictures are reconstructed, or painted-in, using pixel- or patch-based exemplar methods [314], diffusion methods [384], or hybrids of the two [354]. In recent years, however, the advantages of deep-learning-based methods have become prominent. They outclass traditional methods, especially when restoring larger areas, as they are able to learn and consider the semantics of the image. The most common CNN variants for image inpainting are fully convolutional networks and U-Net. However, to avoid filling missing areas with noise and then convoluting this information further, some authors proposed non-standard convolutions. For instance, Liu et al. [223] proposed partial convolution layers that dynamically mask out invalid pixels and cope better with irregular holes. Yu et al. [430] introduced gated convolutions that generalize partial convolutions and provide a learnable dynamic feature selection mechanism across all channels and layers. Similarly, Xie et al. [420] suggested using learnable bidirectional attention maps. In order to better and effectively capture long-distance information, Ning et al. [258] proposed adding a multi-scale attention module. Yan et al. [423] introduced shift connection layers that shift features of known areas to serve as guidance for missing areas and Suvorov et al. [367] presented a combination of Fourier convolutions, a high receptive field perceptual loss, and large training masks for inpainting of large areas. Moreover, many approaches for deep-learning-based inpainting employ one of the various GAN architectures, such as the one proposed by Isola et al. [156], as they feature strong data generation capabilities. Other examples include the works by Shen et al. [335], Yeh et al. [427], and Shao et al. [334]. Similarly, diffusion-based networks such as the one by Rombach et al. [306] achieved impressive results in various image synthesis tasks. These models consist of a hierarchy of denoising autoencoders and can model complex, multi-modal distributions, however, inference tends to be very expensive.

Very recently, transformer networks, originally coming from the natural language processing domain, were discovered to be very effective for computer vision and image processing tasks, such as denoising, too [60]. The main advantage is their ability to model long-range dependencies. Interestingly, Makarov and Borisenko [230] used vision transformers for color-guided depth completion and Li et al. [213] proposed a combination of convolutions and transformers for large hole inpainting. Similarly, Yu et al. [433] presented a bidirectional autoregressive transformer model for diverse inpainting, and Deng et al. [81] designed a transformer model for inpainting with a focus on efficiency. However, transformers are usually still rather slow.

Most research is focused on inpainting color images, and only very few works consider reconstructing depth images. Works that do consider depth images usually are situated in the field of RGB-D reconstruction or lidar-based depth completion and use the color image for guidance. For instance, Ma and Karaman [232] employed a deep regression network to predict depth images based on corresponding color images and sparse depth samples. Fujii et al. [115] used a late fusion GAN to simultaneously reconstruct color and depth images by exploiting the complementary relationship between RGB and depth information. Lee et al. [207] proposed multi-scaled and densely connected locally convolutional layers for depth completion, Tao et al. [370] use a neural network for the prediction of dense depth maps as well as uncertainty estimates, and Jeon et al. [160] performs depth completion based on line features by bridging the conventional and deep learning-based approaches. All these works require color input as well, though. Similarly, Zhang and



Funkhouser [440], as well as Satapathy and Sahay [318], rely on color guidance. In contrast, Jin et al. [163] and Li and Wu [218] presented solutions for depth inpainting without color guidance, however, they are only designed to handle smaller holes. Other works that solely work on depth images can be found in the medical domain, i.e., to reconstruct and in-paint CT or MRI scans. Both, Li et al. [217] and Armanious et al. [10]. for instance, presented promising solutions using patch-based GANs.

For a more comprehensive overview, we refer to the excellent literature review by Zhang et al. [439].

### 3.3.3 Categorization of Depth Errors

Before describing our proposed approach for depth enhancement by depth image completion, we briefly discuss the different types of sensor noise that are typical for TOF sensors and present examples of the main causes that lead to missing depth data. We will explain the noise types using an Azure Kinect as an example TOF sensor, however, they are applicable to other TOF sensors, too. If interested, a more detailed explanation can be found in the Azure Kinect documentation [250].

**Systematic Error** The first error type, called systematic error, comprises the inherent, temporally constant difference between the measured depth and the correct (ground truth) depth. It is defined as:

$$E_{systematic} = \frac{\sum_{t=1}^N d_t}{N} - d_{gt} \quad (3.10)$$

, where  $N$  is for the number of frames used,  $d_t$  is the measured depth at time step  $t$  and  $d_{gt}$  is the ground truth depth. The systematic error can be measured by averaging over several frames to remove other time-varying noise.

**Random error** This error type is caused by shot noise, which is created due to a varying number of photons hitting the sensor each frame, even if the scene and camera are static. Thus, the measured depth will be slightly different for each frame, too. The random error is defined as:

$$E_{random} = \sqrt{\frac{\sum_{t=1}^N (d_t - \bar{d})^2}{N}}, \quad (3.11)$$

where  $N$  is again the number of frames used,  $d_t$  is the depth measured at time  $t$  and  $\bar{d}$  is the average of all depth measurements  $d_t$ .

**Invalidation** In some cases, it is not possible to get valid depth values for some pixels. In that case, pixels are usually invalidated, for instance, by setting them to a depth value of zero. These pixels or areas then appear as holes in the depth images. Invalidation can happen due to different reasons.

**Invalid Signal Strength** One of the reasons for invalidation is too strong infrared signals. Due to the saturation of the pixels, phase information is lost and a depth value can not be calculated. This event can be observed in Fig. 3.35.

In contrast to that, invalidation can also happen when the infrared signal is not strong enough, making it impossible for the depth engine to calculate valid depth values. Figure 3.36 shows such an event.

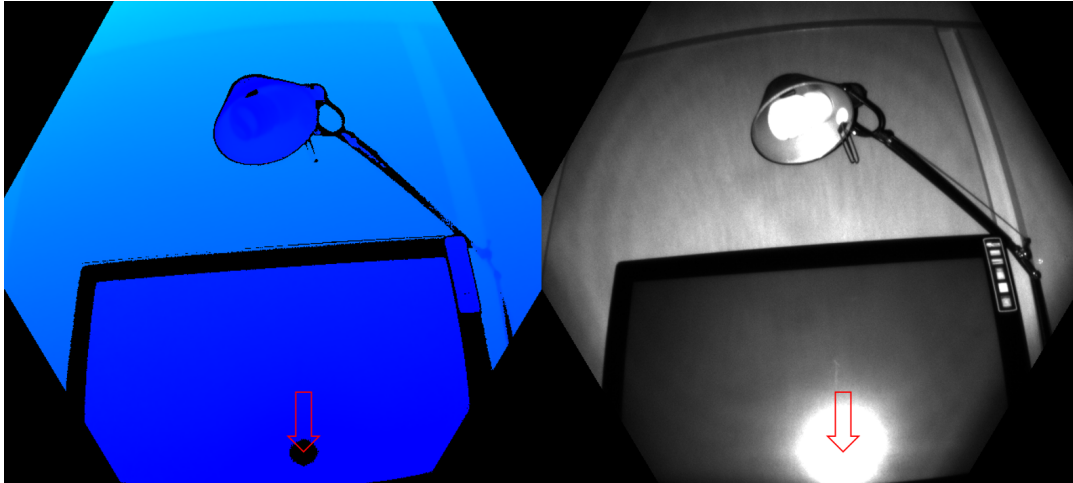


FIGURE 3.35: Invalidated depth values in the depth map (left) due to saturated (too strong) infrared signals in the infrared image (right) [250].

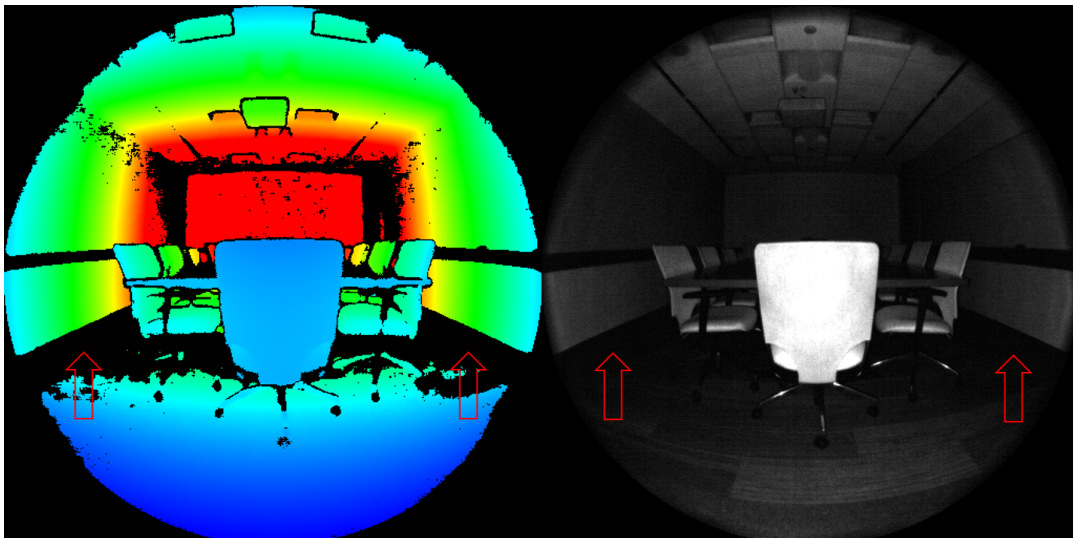


FIGURE 3.36: Invalidated depth values in the depth map (left) due to too weak infrared signals in the infrared image (right) [250].

**Ambiguous Depth** Another reason for depth invalidation is when a pixel receives signals from several objects in the scene, which then leads to an ambiguous depth value. This may occur when the IR light is reflected between objects (multi-path inference). For instance, corners are prone for this to happen, as the IR light may be bouncing off of one wall and hitting the other wall before arriving back at the sensor. That described scenario can be observed in Fig. 3.37.

Ambiguous depth values also commonly happen at object edges, as a pixel may contain a signal from the background and the foreground object. A case like that can be observed in Fig. 3.38.

Generally, the ability to detect valid depth values is highly dependent on the scene geometry (angle between surface and sensor), the object's materials (i.e., reflection, absorption of IR light), and the distance between object and sensor. In the following, we want to specifically consider the challenge of reconstructing these areas with missing depth values.

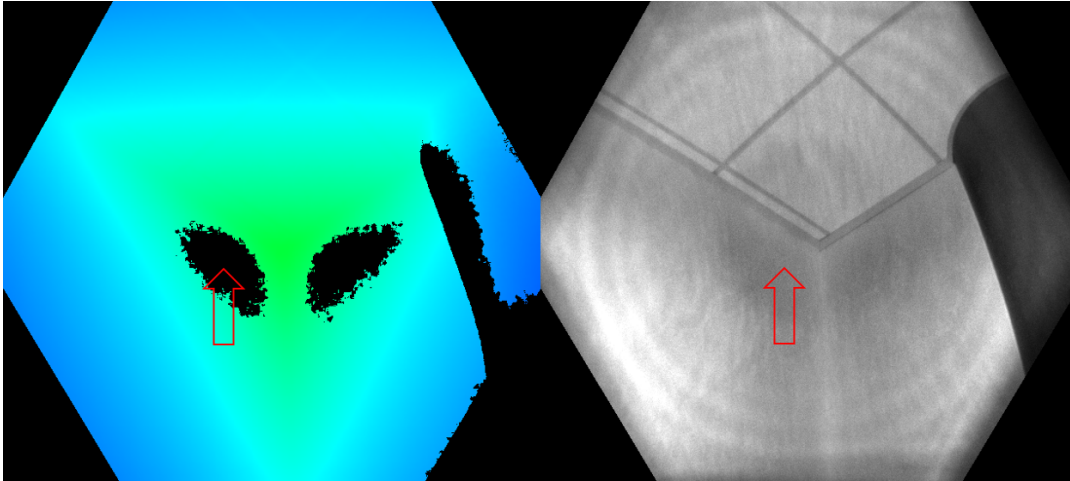


FIGURE 3.37: Invalidated depth values in the depth map (left) caused by multipath inference occurring in a corner (right) [250].

### 3.3.4 Proposed Approach

In order to tackle the issue of (real-time) depth image inpainting, and after thoroughly experimenting with the current state of the art in deep color image inpainting, we decided to adopt two promising works that we considered suitable as a foundation. The first model we chose is the one by Liu et al. that introduced partial convolutions [223], and the second one is the GAN model proposed by Isola et al. [156]. As a baseline for comparison, we also took a standard U-Net model and the more sophisticated state-of-the-art model by Suvorov et al. [367], LaMa, which we expected to be significantly slower, though.

#### Datasets

For the training and evaluation of our models, we resorted to using two publicly available depth datasets, namely, the SceneNet RGB-D dataset by McCormac et al. and the NYU Depth V2 dataset by Silberman et al. The SceneNet dataset provides 5 million photo-realistic RGB-D images of synthesized indoor scenes. We only use the depth images. The images are 16-bit encoded which is similar to real-world input, however, the image resolution is significantly lower than the ones of common depth sensors such as the Azure Kinect. To prevent upsampling artifacts from influencing the training, we use this dataset only for evaluation. The NYUV2 dataset was collected by capturing a wide range of indoor locations within a large city using a Kinect V1 RGB-D camera. Additionally, we created our own custom dataset consisting of mostly static and a few dynamic scenes using the Microsoft Azure Kinect RGB-D camera. As this data lacks a ground truth, we use it only for evaluation, too. In the end, we trained our models with a split of 44984 depth images for the training set, 654 for the validation set, and 5704 for the test set (NYUV2). For the evaluation, we used additional 23 scenes with 6900 images (SceneNet) and 23 scenes with 6739 images (custom dataset).

#### Preprocessing Pipeline

For the training procedure, the images go through a preprocessing pipeline, see Fig. 3.39. First, the images get resized to  $512^2$  and scaled to the range of 0-1 for

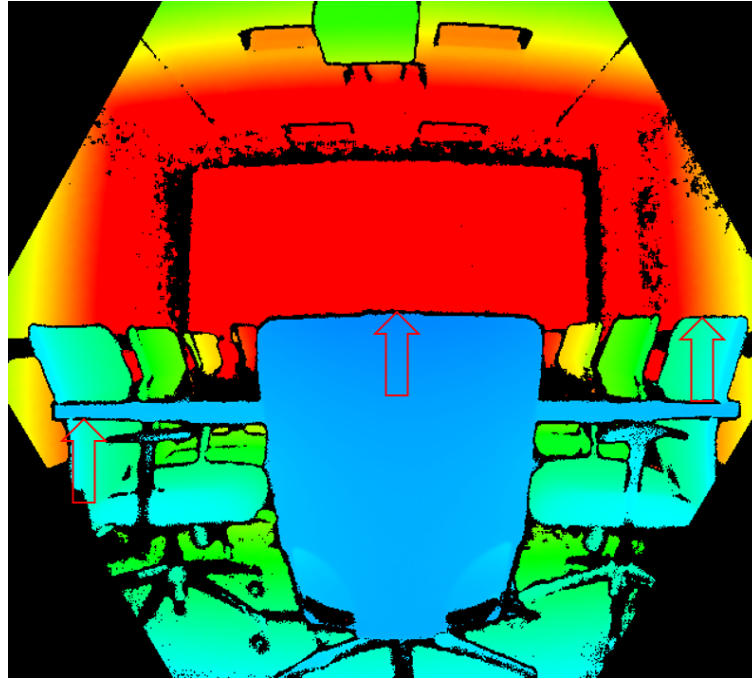


FIGURE 3.38: Depth values that are invalidated because of an ambiguous depth at object edges (foreground and background object) in the depth map [250].

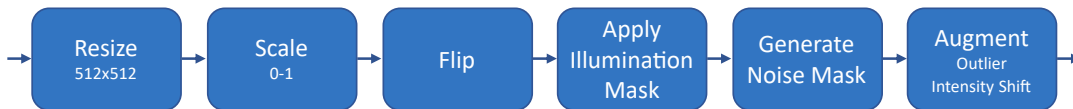


FIGURE 3.39: The preprocessing pipeline. Each image passes through this pipeline during the training process.

compatibility purposes with the models. Next, the images get flipped randomly (90-degree angles). Then, an illumination mask similar to the one of the Azure Kinect is generated and applied to adapt the dataset’s images to real-world input conditions. As the dataset used for training doesn’t contain any holes, we generate synthetic ones using sci-kit-image’s `binary_blob` method. This function outputs a synthetic binary image with random blob-like objects based on certain parameters. Those parameters determine the size of the blobs (`blob_size_fraction`) and the fraction of image pixels that are covered by the blobs (`volume_fraction`). The output of the function is a matrix of boolean values, where the generated blobs have the value “true” and everything outside of the blobs is set to “false”. Applied to our dataset images by element-wise multiplication, all areas not covered by the binary blobs get invalidated. In order to generate more realistic noise, we combine multiple random masks with different scales and frequencies, see Fig. 3.40. The idea behind that is to create a first blob image, that creates blobs of bigger sizes so that the image still has big valid areas after invalidation that can serve as a reference for the inpainting process. A second blob image is generated to create additional small-sized areas within the areas that would have been invalidated using only the first blob image. As a way to imitate the grain-like noise that is present in Azure Kinect images, very small blobs are created in the third blob image. To guarantee a diverse input, the final noise masks are evenly drawn from multiple categories with varying percentages of invalid pixels and sizes of holes, see Fig. 3.41. An additional advantage of having

these multiple noise categories is that they allow us to compare the performance of the networks on different noise categories in the evaluation. Specifically, we decided to create six different categories with the hole-to-image area ratios of:  $(0.01, 0.1]$ ,  $(0.1, 0.2]$ ,  $(0.2, 0.3]$ ,  $(0.3, 0.4]$ ,  $(0.4, 0.5]$ , and  $(0.5, 0.6]$ . The noise categories are created by changing the volume\_fraction parameters for the three to-be-fused binary blob images according to the needed hole-to-image ratio. The exact values can be read in Table. 3.5. The blob\_size\_fraction parameter stays the same for the three blob images over all categories: 0.1 for big blobs, 0.05 for small blobs, and 0.01 for grainy blobs. These parameters were estimated empirically.

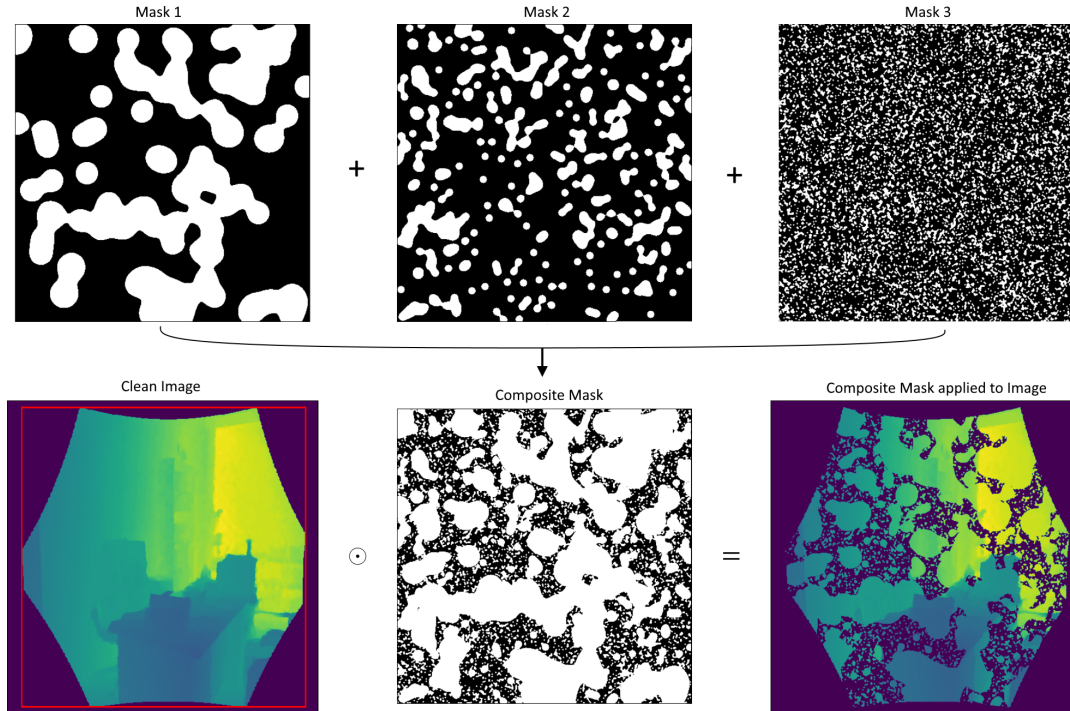


FIGURE 3.40: Noise mask generation process. Three masks are created using sci-kit-image’s binary\_blob method and fused together into a composite mask. That composite mask is then applied to a depth image to invalidate values according to the mask. The size of the mask is slightly smaller than the depth image size and is indicated by the red square.

Category	Mask1	Mask2	Mask3
$(0.01, 0.10]$	0.40	0.70	0.70
$(0.10, 0.20]$	0.40	0.50	0.50
$(0.20, 0.30]$	0.40	0.35	0.35
$(0.30, 0.40]$	0.40	0.25	0.25
$(0.40, 0.50]$	0.40	0.15	0.10
$(0.50, 0.60]$	0.40	0.03	0.03

TABLE 3.5: Values for the volume\_fraction parameters of the three binary blob masks for each mask category.

Finally, we adopt other classical data augmentation techniques, in addition to the previous flipping of the images, by applying homogeneous intensity shifts and artificially generated outliers. We do generate the latter, as we observed pixels that

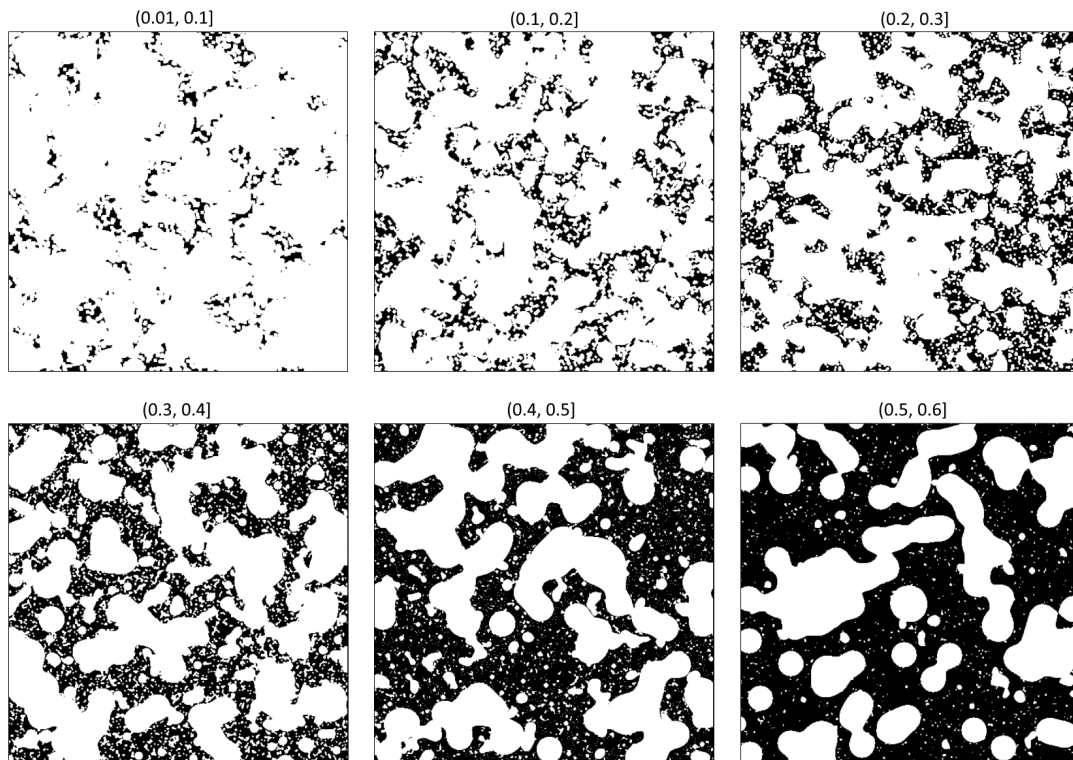


FIGURE 3.41: Noise masks from six categories with varying percentages of invalid pixels and sizes of holes that can be generated in the mask generation process.

are of higher intensity than the neighboring pixels (referred to as outliers) in our custom dataset. We synthetically generate these outlier pixels for the training dataset, as those are not present in the NYUV2 dataset. The creation of those outlier pixels is done according to Algorithm 3.

---

**Algorithm 3** Process of outlier pixel generation

---

```

while outlier count not reached do
  get random pixel position inside of the illumination mask
  if not already an outlier pixel then
    mean = mean pixel value of neighborhood
    range = range from mean plus margin to maximal pixel value
    sample random value from range
    add random value to mean as new pixel value
    increase outlier count

```

---

First, a random  $x$  and  $y$  coordinate is sampled from the coordinates of the octagonal illumination mask. Then the mean pixel value of the neighboring pixels is calculated. The neighborhood is picked to be the pixels that are direct neighbors, also counting diagonal neighbors. A range is then calculated between the mean value of the neighborhood, including a margin, and the maximal possible pixel value of the encoding. The margin is a fixed value based on the observed outliers in the custom dataset. The range is then used to sample a random value that will be added to the pixel's original value at the selected position. This creates outlier pixels that have greater values than the neighbors, as seen in Fig. 3.42.

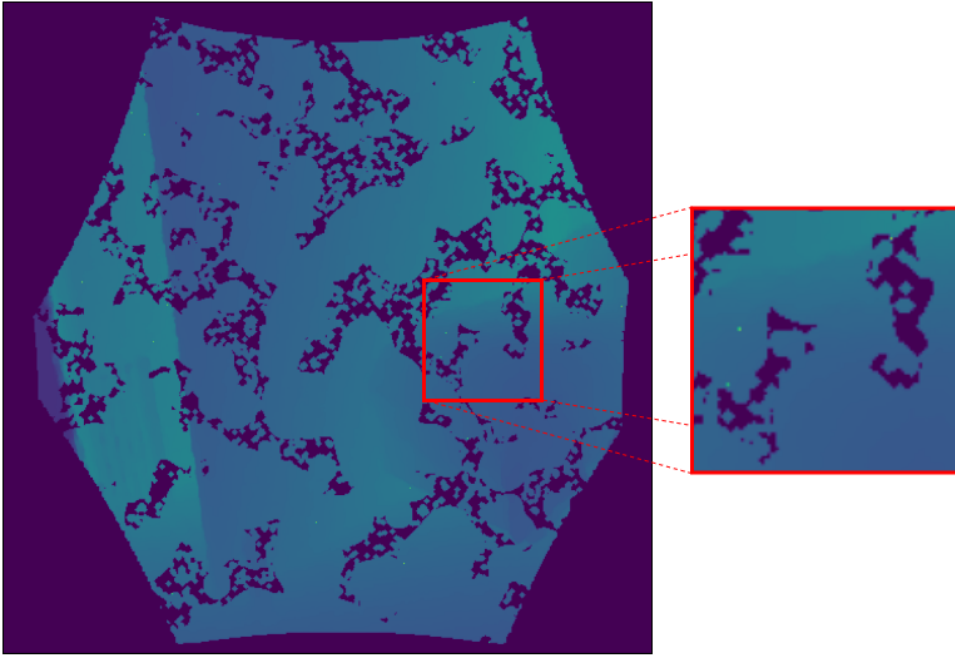


FIGURE 3.42: Generated outliers in the augmentation step. The image on the right shows three generated outliers in a close-up. The three outliers are recognizable by their strong intensity, compared to the neighbor pixels.

### Network Details

In the following, the network details of our models get briefly described. For more details, we refer to the corresponding original papers.

**Partial Convolution** Our first network model is based on the one presented by Liu et al. [223] and, like the original, follows a U-Net architecture with partial convolution layers. As a small recap, this method uses a custom convolutional layer that is conditioned to only use valid pixels for the prediction of missing pixels. The information of valid and invalid pixels is passed to the layer for the separation process in the form of a binary mask. That binary mask is updated in the layer after each partial convolution. The partial convolutional layer thus consists of a partial convolution and an internal mask update operation. Additionally, a new loss function made up of different weighted parts, was introduced by the authors. The advantage of this method is, that it is an end-to-end method, where an image is given to the network and the output is an inpainted image, that does not need any further postprocessing. In contrast to other methods, where the noise is either set to a predefined region or has a certain shape, this method can be used to inpaint holes/noise of irregular and arbitrary shapes, which is a perfect precondition for the incomplete depth images produced by the Azure Kinect. We decided to use the partial convolutional layer, like Liu et al., in a U-Net architecture, as the simplicity of the architecture makes it possible to apply the network to bigger images in a reasonable amount of time. This could potentially also enable the inference of images in real-time, which will be further investigated in the result section. Furthermore, training the U-Net does not require a huge amount of data. This was proven in the initial publication of the architecture by Ronneberger et al. [307], which is a benefit due to our training set

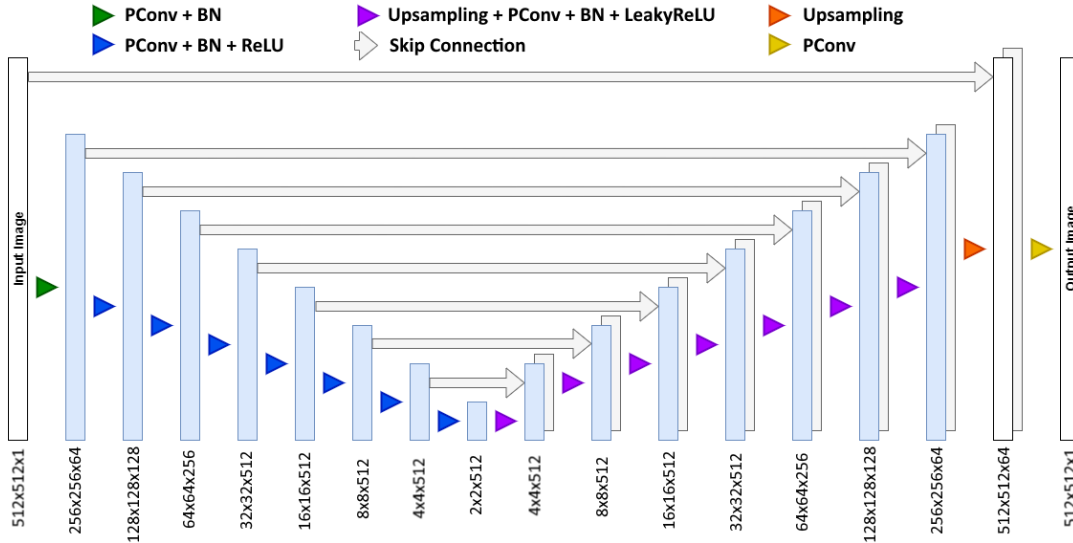


FIGURE 3.43: Network architecture of our partial convolution U-Net model, which closely follows the one by Liu et al. [223].

size. In contrast to Liu et al., our model features only one input and output channel, respectively, though. The complete network structure is depicted in Fig. 3.43.

We chose an input resolution of  $512^2$ , as it is the closest square number to the resolution of the Azure Kinect images. The kernel sizes for the partial convolutions in the encoder part are 7,5,5,3,3,3,3 and 3, following the presented layer order. The decoder uses filter sizes of 3 for all convolutions. For all convolutions in the network, stride values of 2 are used. The implementation of this network was done using PyTorch 1.10.1 and the existing third-party implementation of Ryan WONGSA for the U-Net architecture [415] and loss functions. However, adjustments were made due to the fact that the crucial weight initializations as well as the input normalizations of the VGG-16 network were missing. Moreover, the implementation of the partial convolutional layer from the original authors was used [222], too.

**GAN** Our second network model is based on the GAN architecture presented by Isola et al. [156] that uses a U-Net for the generator and a convolutional PatchGAN classifier for the discriminator. The latter penalizes structure at the scale of image patches. The idea behind this architecture is that low-frequency correctness regarding ground truth input can be accurately forced using a classical L1 loss term in the generator. Then, the discriminator has to only model the high-frequency structures, which are usually confined to small local regions, instead of analyzing the image as a whole. The advantage of this approach is that the network requires fewer parameters that have to be optimized, runs faster, and can be applied to arbitrarily large images. The generator part of the GAN is very close to the architecture presented in Fig. 3.43: The encoder consists of 8 identical blocks instead of 7, which are Conv-BN-LeakyRelu blocks that use the same filter sizes of 64, 128, 256, 512, 512, 512, 512. The decoder consists of seven Upsampling-Concat-BN-Relu blocks. Additional dropouts of 50% are applied to the first three blocks after the normalization process. A final convolution maps the number of output channels. The input dimensions are  $512^2 \times 3$ , as three depth images are stacked. All convolutions of the network use filters of size 4 with a stride of 2. The discriminator consists of one Conv-LeakyRelu



layer followed by 3 Conv-BN-LeakyReLU blocks and a single Conv-ZeroPadding-Sigmoid block. This outputs a  $30^2$  image patch that can classify a  $70^2$  portion of the input image. The implementation was done using TensorFlow 2.6.0.

**Convolutional U-Net** As a baseline for comparison of the previous models, we also utilize a convolutional neural network with a standard U-Net architecture, although models with normal convolutional layers that treat all image pixels the same and even share filter weights are not ideal for image inpainting. The network architecture follows the exact architecture presented in Fig. 3.43 with the only difference that all partial convolutional layers were swapped with regular convolutions.

**LaMa:** To get a more complete picture and to compare the models with more sophisticated networks, we also adopted the LaMa network by Suvorov et al. [367]. It is specifically designed for the inpainting of large areas by using fast Fourier convolutions that provide a large receptive field, as well as an adapted perceptual loss and large training masks. However, as it is more complex, we expect it to be significantly slower and possibly not real-time capable. The premise of the work is, that in order to effectively fill in large holes, especially early on in the network, layers with a wide receptive field are crucial. Thus, they employ (nine) Fast Fourier convolution blocks that consist of two parallel branches, a local one with regular convolutions and one using Fast Fourier transformations for global context. For details about the architecture, we refer to the original paper, from which we directly adopted it.

### Training Procedure

Liu et al. compared their U-Net architecture with partial convolutional layers against two GAN architectures in their pre-trained versions, without retraining them on the same dataset and loss terms as the U-net. Therefore, the comparison was not completely fair. However, GANs generate results that are coherent in texture and structure. For that reason, in contrast to Liu et al., we explicitly (re)trained our employed GAN architecture for the depth image inpainting task, which gives a fair and strong comparison to the other methods.

The partial convolution, U-Net and GAN model were trained for 7 epochs using a batch size of 2, due to the huge memory load. As loss function for all three models, we used, similarly to the partial convolution paper by Liu et al., a weighted combination consisting of two per-pixel accuracy losses, a perceptual loss, two style losses, and a total variation loss. For the individual equations and a detailed description, we refer to the original paper. In short, per-pixel accuracy between the inpainted image and the target image is forced by the L1 loss. Here, the L1 loss is split in two: The first one focuses on only the valid pixels and the second one compares the inpainted hole pixels with the target pixel values. The perceptual loss measures how similar the high-level features/content of the inpainted image are compared to the ground truth. On the other hand, the style losses measure how similar in style the images are. This is done by calculating the difference in correlation between the feature maps of a given layer. The style loss is calculated on the direct output image as well as on the composite image of the to-be-inpainted image together with the predictions for the hole pixels. Lastly, the total variation loss is used to ensure spatial continuity and encourage smoothness in an image. The resulting total loss is therefore

$$\mathcal{L}_{total} = \mathcal{L}_{valid} + 6\mathcal{L}_{hole} + 0.05\mathcal{L}_{perceptual} + 120(\mathcal{L}_{style_{out}} + \mathcal{L}_{style_{comp}}) + 0.1\mathcal{L}_{tv} \quad (3.12)$$

We experimented with different weights but found the ones used in the paper to be the best-performing ones. In the case of the GAN model, the generator loss is a combination of the previous total loss and the original generator loss as described in the paper by Isola et al. The resulting loss function for the GAN method is defined as

$$\mathcal{L}_{GAN} = \mathcal{L}_{gen} + \lambda \mathcal{L}_{total}, \quad (3.13)$$

where  $\mathcal{L}_{gen}$  is the loss given by the generator,  $\mathcal{L}_{total}$  is the total loss described in Equation 3.12 and  $\lambda$  is a weighting factor. We set its value to 100, as done by Isola et al., due to the fact that the method creates artifacts without or with a lower weighting value.

For LaMa, we employed a different approach and directly adopted its original loss function as well as the pre-training it went through. We did this as this model is a popular state-of-the-art model and we aimed at keeping the comparability with other works as high as possible and the proposed loss function is a crucial part of the model. In fact, it consists of multiple terms, namely, the high-receptive field perceptual loss (that is based on a pre-trained base network with a fast-growing receptive field), an additional gradient penalty term, as well as an adversarial and a discriminator-based perceptual loss, which, together, should provide plausible local details. Eventually, we additionally trained it with a batch size of 5. The best-performing epoch, which we selected for evaluation, was the 5th.

### 3.3.5 Results

The evaluation of all models was done using an Intel Core i5-10400F CPU, 16 GB of RAM, and an NVIDIA GeForce RTX 2070.

**Training Results** To evaluate the models' progress during training, we measured and plotted the individual loss term values for each batch, see Fig. 3.44 (left). For convenience, we abbreviate the models' names with Conv, PConv, and GAN. As the loss values vary drastically between batches because of the inhomogeneous hole characteristics, we smoothed the loss curves using an exponential weighted moving average. The PConv method already converges after the first epoch and then mostly stagnates. This may be due to the fact that U-Nets are good at propagating the global image context via skip links, leading to the valid pixels being estimated with a good performance from the start. The style loss term seems to be comparably high though. In comparison, the Conv method has worse style loss values but similar results for the remaining loss terms. The GAN method shows the same performance as the Conv method, resulting in almost identical loss values for all loss terms. Additionally, the generator loss gives an insight into the minimax game of the generator and discriminator. The high values indicate that the discriminator must have been good at distinguishing real and generated images throughout the training, which in turn means that the generated images may still incorporate elements that make them look fake.

We also tracked the models' training performance by measuring the mean loss values of the validation phases that occurred after each epoch, see Fig. 3.44 (right). The performance of the PConv method on the validation set shows similar results to the training set performance. However, the style loss values seem to be slightly lower on the validation set, which brings down the total loss values. Generally, this could indicate that the model is under-fitted. The same behavior can be observed in the loss curves of the Conv method, although, here, the L1-hole loss is higher

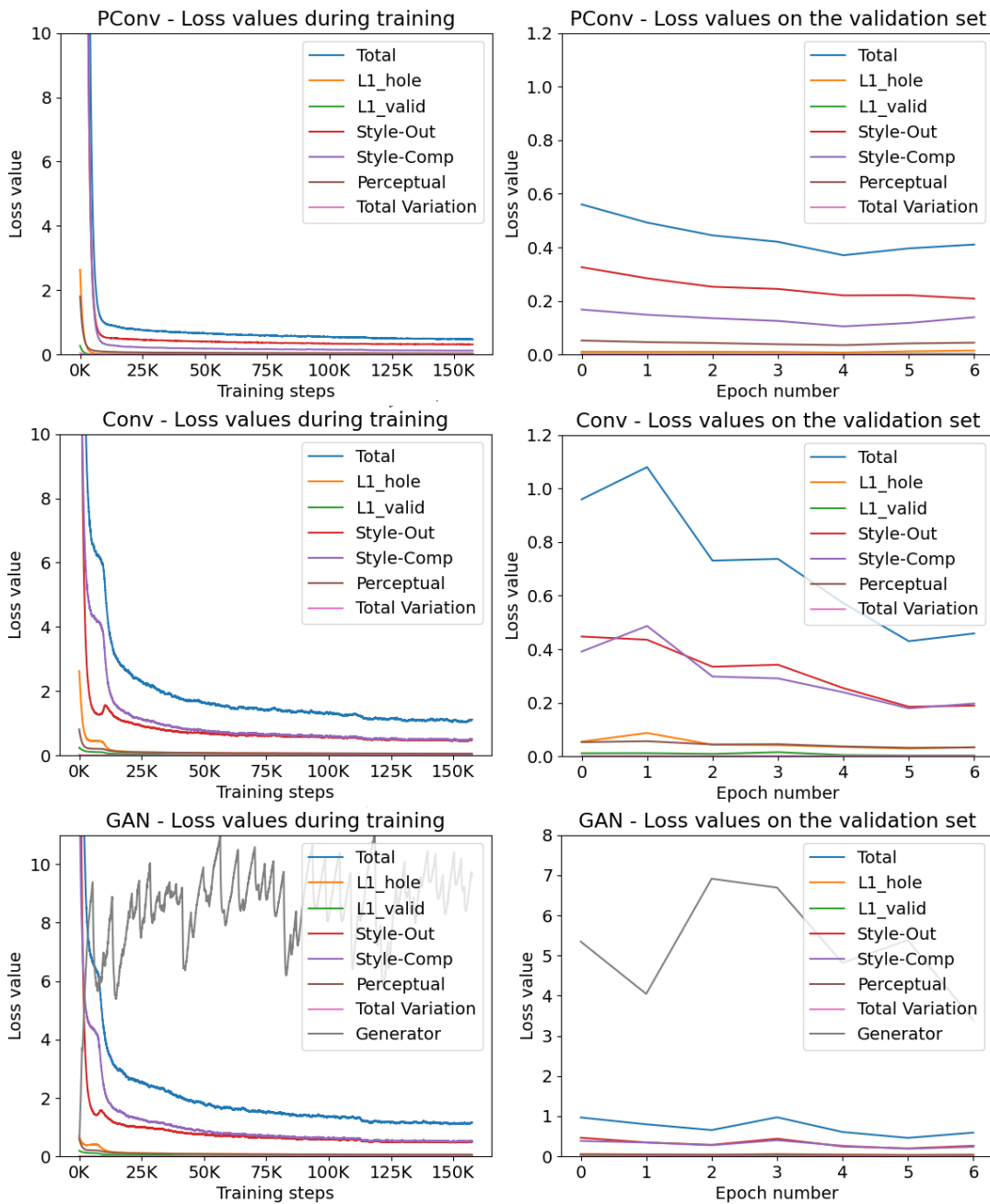


FIGURE 3.44: Left: Progress of loss terms during training. Right: Mean loss term values over validation set.

than the respective loss of the PConv method, which may suggest that it is better at filling in holes. The same loss term is also lower for the GAN method. Following the potential underfitting phenomenon of the other two methods, the GAN method shows the same behavior.

**Inference Timings** First, we measured the duration of inference needed for inpainting a  $512^2$  depth image. A fast inference is crucial for practical real-time applications, i.e., as a preprocessing step in a longer pipeline. As depth sensors usually capture with 30 Hz, the inference time must stay below 33 ms for real-time use. To replicate a data stream of images, the images were inpainted one after another, instead of as a batch. For the GAN method, we measured a pure inference time of 24.3 ms, for the Conv method 24.93 ms, and for the PConv method 9.37 ms. Including preprocessing, we get 27.69 ms, 26.29 ms, and 34.34 ms, respectively. The PConv model takes the longest for the preprocessing as it needs more steps than the other models, i.e., an extra input mask. However, the time for pure inference is the quickest. Generally, even though there is still potential for optimization, these models are quick enough for real-time application. In contrast, LaMa takes 60.02 ms and, thus, is significantly slower and not quite real-time capable. Out of interest, we also tested a diffusion-based model [306] but, as expected, the inference was extremely slow with 3-4 seconds for an image with 50 sampling steps (which was, as we found, a “sweet spot” for image legibility and speed). Unfortunately, the inpainting results were still comparatively poor. And although better output quality can be achieved with more sampling steps during inference, doing so only impacts inference time even more, which is why we did not consider latent diffusion-based models further.

**Quantitative Results** To quantitatively evaluate the performance of our models, we calculated and compared the mean absolute error (MAE), mean squared error (MSE), PSNR, and structural similarity index measure (SSIM) on the test sets of the NYUV2 and SceneNet RGB-D datasets (only depth used). Moreover, we separately computed the metrics for the different hole/mask categories, which bundle images with similar ratios of valid/invalid areas to get more detailed insights.

The results on the NYUV2 dataset show that LaMa consistently performs best. Moreover, we see a better performance of the GAN method on the first four mask categories, especially if looking at the MAE and MSE, see Table 3.6. The performance gradually decreases with each category, though, and after the fourth category, the PConv method overtakes the GAN performance in terms of SSIM and PSNR values. In comparison, the Conv method is (as expected) the worst-performing one. Generally, the PConv method seems to be the most consistent method and better at dealing with bigger holes than the GAN and Conv methods.

Overall, the models seem to perform similarly on the SceneNet RGB-D dataset as on the NYUV2 dataset (see Table 3.7): For the lower mask categories, the GAN method outperforms the Conv and PConv methods, while the PConv method shows better results on the higher categories, and is the most consistent one overall. LaMa again performs most often the best. However, in terms of SSIM, here, GAN/PConv perform better.

**Qualitative Results** After the quantitative evaluation, we did a qualitative evaluation of the inpainting performance based on a selection of test images from different mask categories. This evaluation is, naturally, subjective but possibly also more reliable. Fig. 3.45 shows the results using the NYUV2 dataset. For all three mask

TABLE 3.6: Inpainting results on the NYUV2 test set using the six hole categories (percent of invalid pixels; more/bigger holes to the right). The best value per block is marked in bold. LaMa always performs best. The GAN method performs second best on smaller mask categories while the PConv method performs second best on bigger ones and produces the most consistent results.

Metric/Method	(0.01,0.10]	(0.10,0.20]	(0.20,0.30]	(0.30, 0.40]	(0.40, 0.50]	(0.50,0.60]
MAE/PConv	4.89	5.24	5.10	5.32	5.79	7.61
MAE/Conv	3.53	3.24	3.27	3.52	5.64	13.35
MAE/GAN	1.79	1.77	1.93	2.46	4.48	11.18
MAE/LaMa	<b>0.06</b>	<b>0.18</b>	<b>0.31</b>	<b>0.42</b>	<b>0.64</b>	<b>1.00</b>
MSE/PConv	47.82	54.00	54.21	60.67	77.99	154.54
MSE/Conv	62.90	56.49	58.45	69.16	131.46	612.88
MSE/GAN	6.79	7.34	10.93	16.68	67.83	415.20
MSE/LaMa	<b>0.28</b>	<b>0.87</b>	<b>1.67</b>	<b>2.48</b>	<b>4.81</b>	<b>12.18</b>
PSNR/PConv	35.12	34.81	34.70	34.43	33.27	30.47
PSNR/Conv	32.29	32.40	32.13	31.64	28.59	22.40
PSNR/GAN	41.42	40.51	38.94	37.01	31.13	23.72
PSNR/LaMa	<b>55.04</b>	<b>50.15</b>	<b>47.38</b>	<b>45.74</b>	<b>43.01</b>	<b>39.22</b>
SSIM/PConv	0.9799	0.9771	0.9746	0.9701	0.9630	0.9385
SSIM/Conv	0.9344	0.9230	0.9184	0.9026	0.8819	0.8264
SSIM/GAN	0.9935	0.9874	0.9815	0.9759	0.9480	0.8814
SSIM/LaMa	<b>0.9987</b>	<b>0.9966</b>	<b>0.9943</b>	<b>0.9927</b>	<b>0.9898</b>	<b>0.9842</b>

TABLE 3.7: Inpainting results on the SceneNet RGB-D test set (depth only). Like in Tab. 3.6, LaMa performs most often best, the GAN method performs second best on smaller mask categories while the PConv method performs second best on bigger ones and produces the most consistent results.

Metric/Method	(0.01,0.10]	(0.10,0.20]	(0.20,0.30]	(0.30, 0.40]	(0.40, 0.50]	(0.50,0.60]
MAE/PConv	66.89	81.85	77.27	65.85	63.67	110.62
MAE/Conv	118.68	103.84	101.80	113.30	182.04	438.79
MAE/GAN	66.49	65.07	72.18	89.61	176.30	414.24
MAE/LaMa	<b>4.28</b>	<b>10.52</b>	<b>16.09</b>	<b>20.75</b>	<b>30.77</b>	<b>45.54</b>
MSE/PConv	21732	23122	21622	18157	<b>16787</b>	32051
MSE/Conv	73128	66152	66116	78239	133677	669006
MSE/GAN	8414	9023	13010	21732	95940	588678
MSE/LaMa	<b>5044</b>	<b>8187</b>	<b>9521</b>	<b>10858</b>	17161	<b>29519</b>
PSNR/PConv	38.83	39.08	39.18	38.91	37.85	35.58
PSNR/Conv	35.50	35.89	35.77	35.41	32.41	26.33
PSNR/GAN	44.37	43.76	42.38	40.34	34.32	26.90
PSNR/LaMa	<b>57.31</b>	<b>53.02</b>	<b>50.59</b>	<b>49.23</b>	<b>46.93</b>	<b>43.82</b>
SSIM/PConv	0.9881	0.9876	0.9867	<b>0.9866</b>	<b>0.9855</b>	<b>0.9818</b>
SSIM/Conv	0.9659	0.9606	0.9556	0.9510	0.9388	0.8919
SSIM/GAN	<b>0.9960</b>	<b>0.9931</b>	<b>0.9885</b>	0.9834	0.9674	0.9166
SSIM/LaMa	0.9958	0.9899	0.9855	0.9829	0.9793	0.9755

categories, LaMa produces the best results that are very close to the original. The PConv method is also able to create good results without apparent visual artifacts, apart from a slight blur in the last row with a mask of 40%-50% hole-to-image ratio. For the GAN method, the results for the small mask are very close to the ground truth image. However, on the medium and big masks, we can see slight deteriorations and then even more artifacts occurring, respectively. The Conv method visibly leads to the worst results throughout all mask categories, as can be seen by the increased blurriness and other (dark, cloudy) artifacts. Generally, we find that the qualitative results are consistent with the quantitative ones.

Looking at the inpainting results using the SceneNet RGB-D dataset in Fig. 3.46, we come to similar conclusions, i.e., that LaMa performs better than all the others, especially for the bigger mask categories. The PConv method creates reasonably good results for the small and medium mask categories, the GAN performs well in the small category, and the Conv method is the worst-performing method. However, on this dataset, all methods except for LaMa have issues with artifacts in the form of too-bright or too-dark areas that get more severe with bigger masks. This phenomenon could possibly be because of systemic differences between this dataset and the one used for training (NYUV2). For instance, this dataset with synthetically created images generally has sharper edges and objects than the NYUV2 dataset, which also incorporated errors that slightly degrade the images.

In order to evaluate our models on real-world data, we first investigate the effects of the inpainting methods on the valid areas. Ideally, they should remain unchanged. As can be seen in Fig. 3.47, which shows the color-coded deltas between the original and inpainted images, this is mostly not the case. The PConv method leads to relatively small differences, mostly along the edges of objects, corners, or at thin shapes. This could be an effect of the model trying to prevent hard edges and instead favoring slow transitions. The GAN method performs better at far corners and edges and, generally, produces images with more even deltas. Moreover, it creates the sharpest results with more abrupt object transitions. An odd issue with the GAN method is the distinct artifacts that occur consistently in the upper right corner. We suspect this to be an issue with the value of the introduced weighting factor  $\lambda$  for the loss function, as the authors of the original method suggested that lower values lead to sharper results but, in turn, lead to more artifacts. The Conv method, again, leads to the worst results and produces the biggest deltas throughout the whole image. Interestingly, in contrast to the others, LaMa does not change the originally valid areas at all, which is the best result.

For a final comparison of the models, we compare the resulting images after inpainting, again, using our own custom dataset. As visible in Fig. 3.48 and Fig. 3.49, all methods are able to create reasonable predictions for the missing areas, although the Conv method produces more blurry results. Interestingly, PConv and LaMa as well as Conv and GAN tend to have a similar behavior. Generally, LaMa tends to create the most plausible and visually pleasing results, followed by PConv. However, one drawback of these methods seems to be the prediction around outlier pixels. The advantage of LaMa on this real-world dataset is smaller as with the other datasets though. Moreover, in some cases, the GAN method produces better results, hence, there seems to be no method that is categorically superior.

**Ablation Study** To investigate the effects of the individual loss terms, we trained our partial convolution model (PConv) while switching off one loss term at a time, e.g., no perceptual-/style-/total variation(TV)/valid loss). While Liu et al. [223] found that removing the style- or total variation losses leads to significantly worse

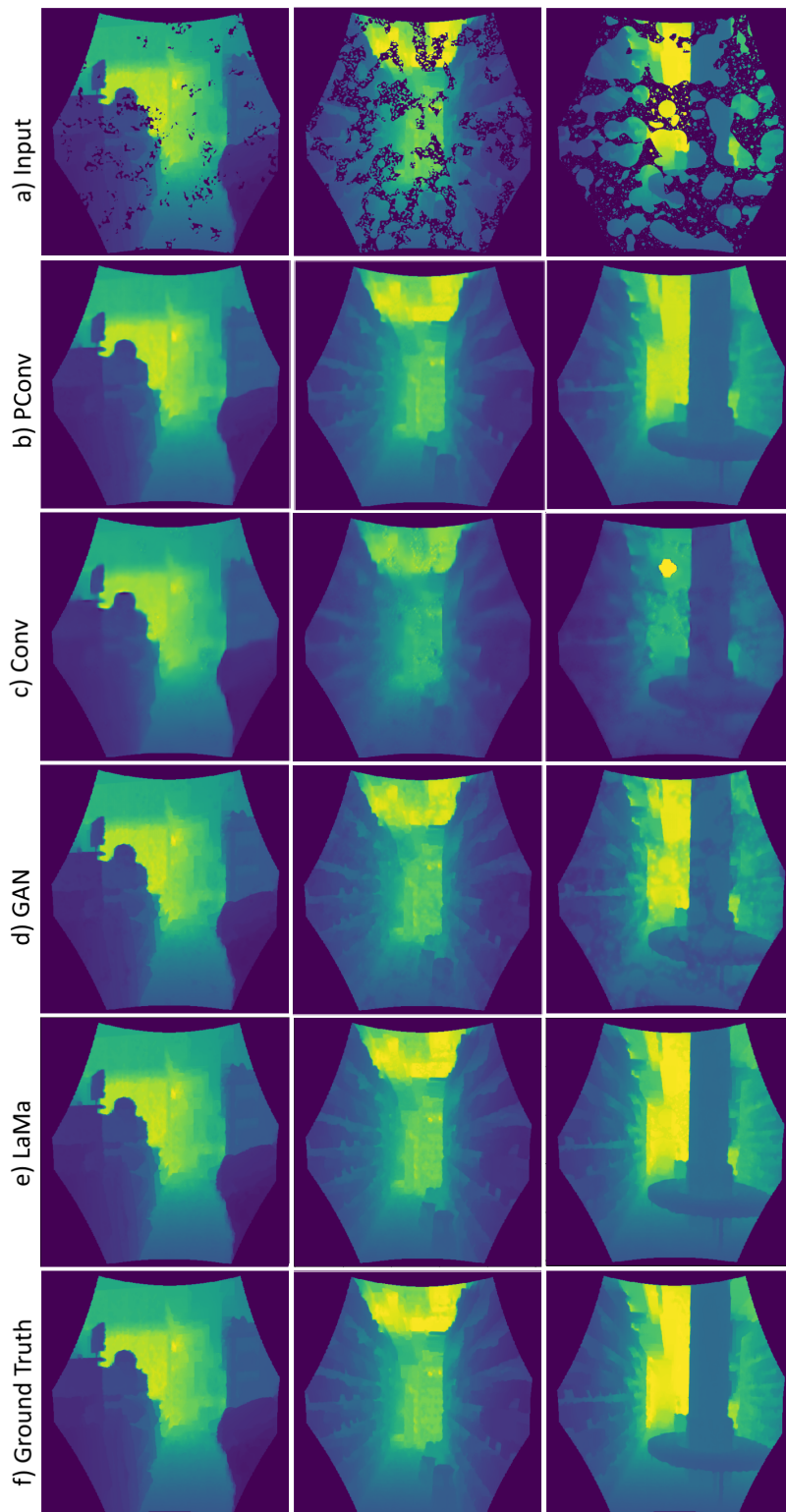


FIGURE 3.45: Visual inpainting results on the NYUV2 test set using various hole categories (columns). LaMa performs best, the PConv method performs second best, the GAN struggles with bigger holes, and the Conv method is the worst.

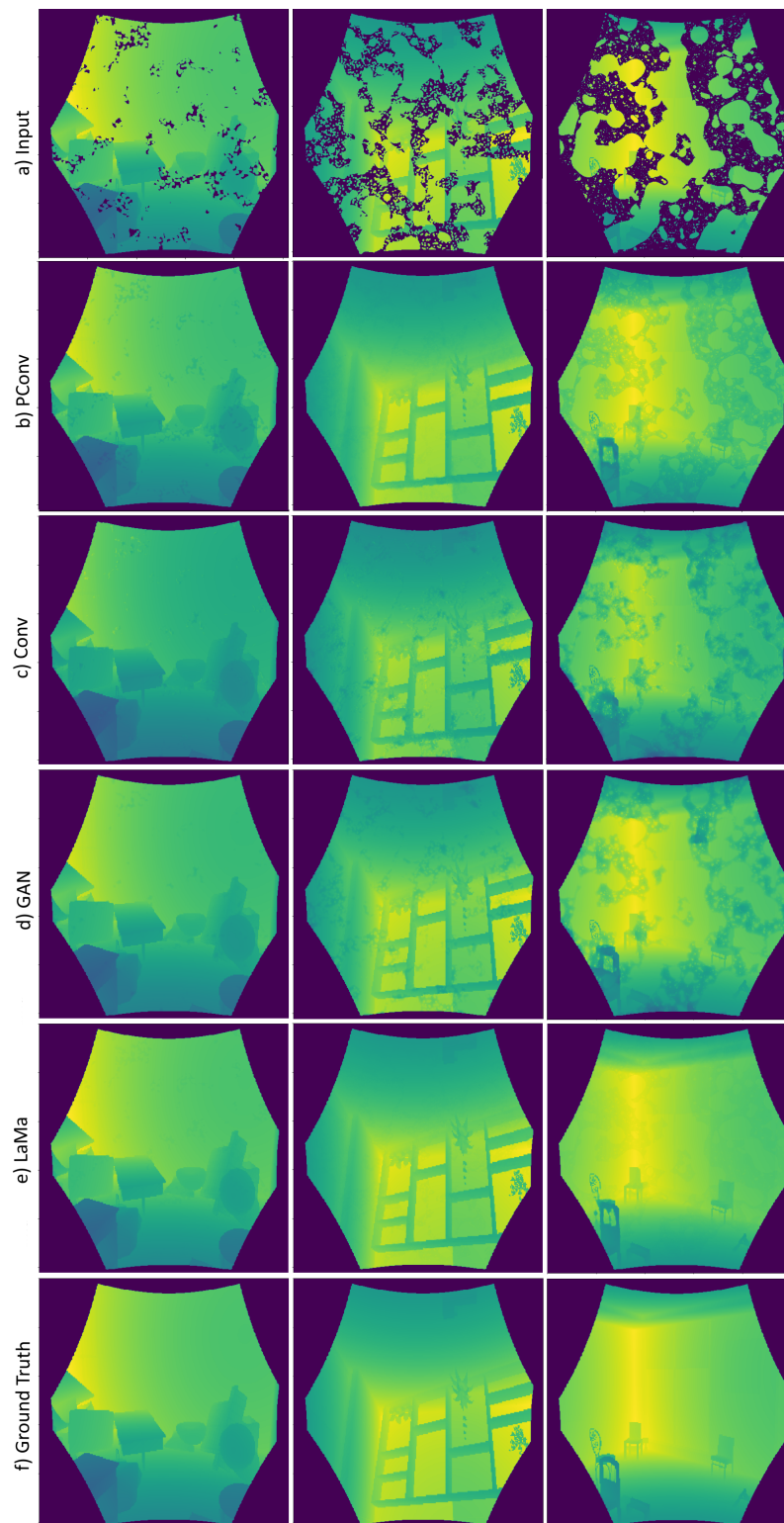


FIGURE 3.46: Visual inpainting results on the SceneNet RGB-D test set (depth only) using various hole categories (columns). All methods except for LaMa, which performs best, produce distinct artifacts. However, PConv and GAN perform reasonably well in the medium/small categories, and Conv is again the worst-performing method.



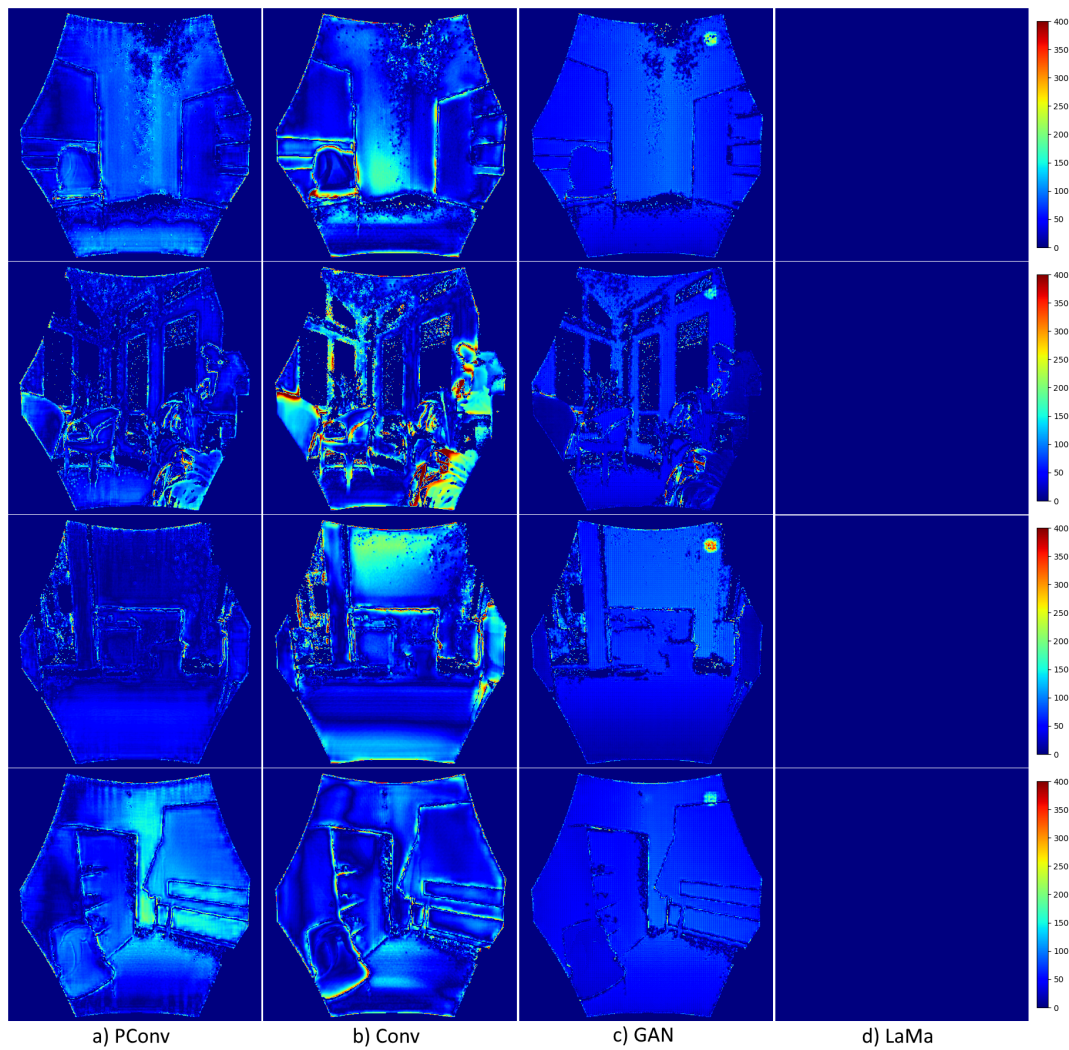


FIGURE 3.47: Color-coded pixel-wise deltas of originally valid areas after inpainting using our own dataset. The holes were reintegrated from the input data. The Conv method alters the original data around holes the most (for smoother transitions), the GAN the least, and LaMa not at all.

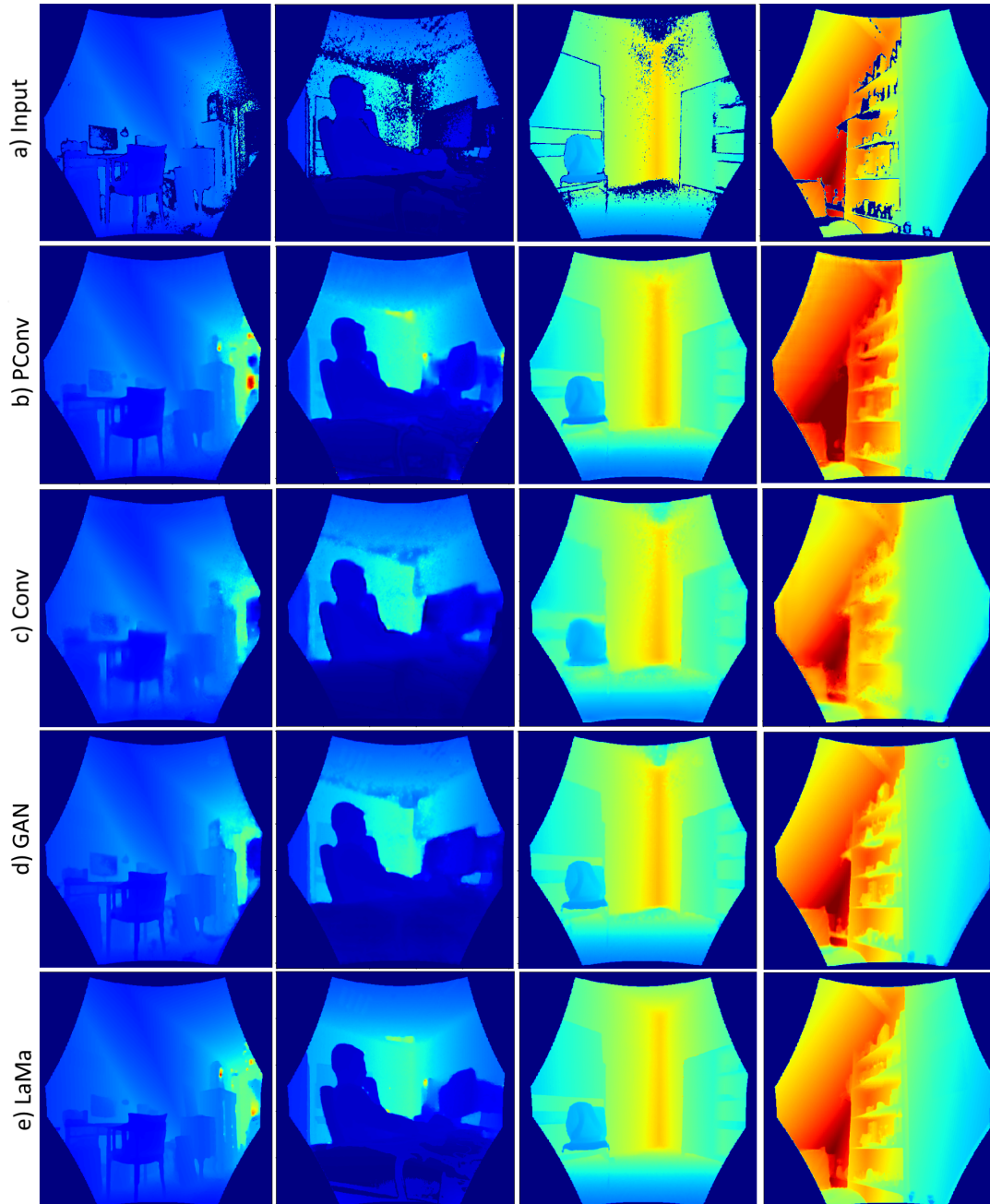


FIGURE 3.48: Inpainting results with our own dataset. The LaMa method most often produces the best visual results. PConv behaves quite similar, and both struggles with outliers. However, in some cases, the GAN performs the best.

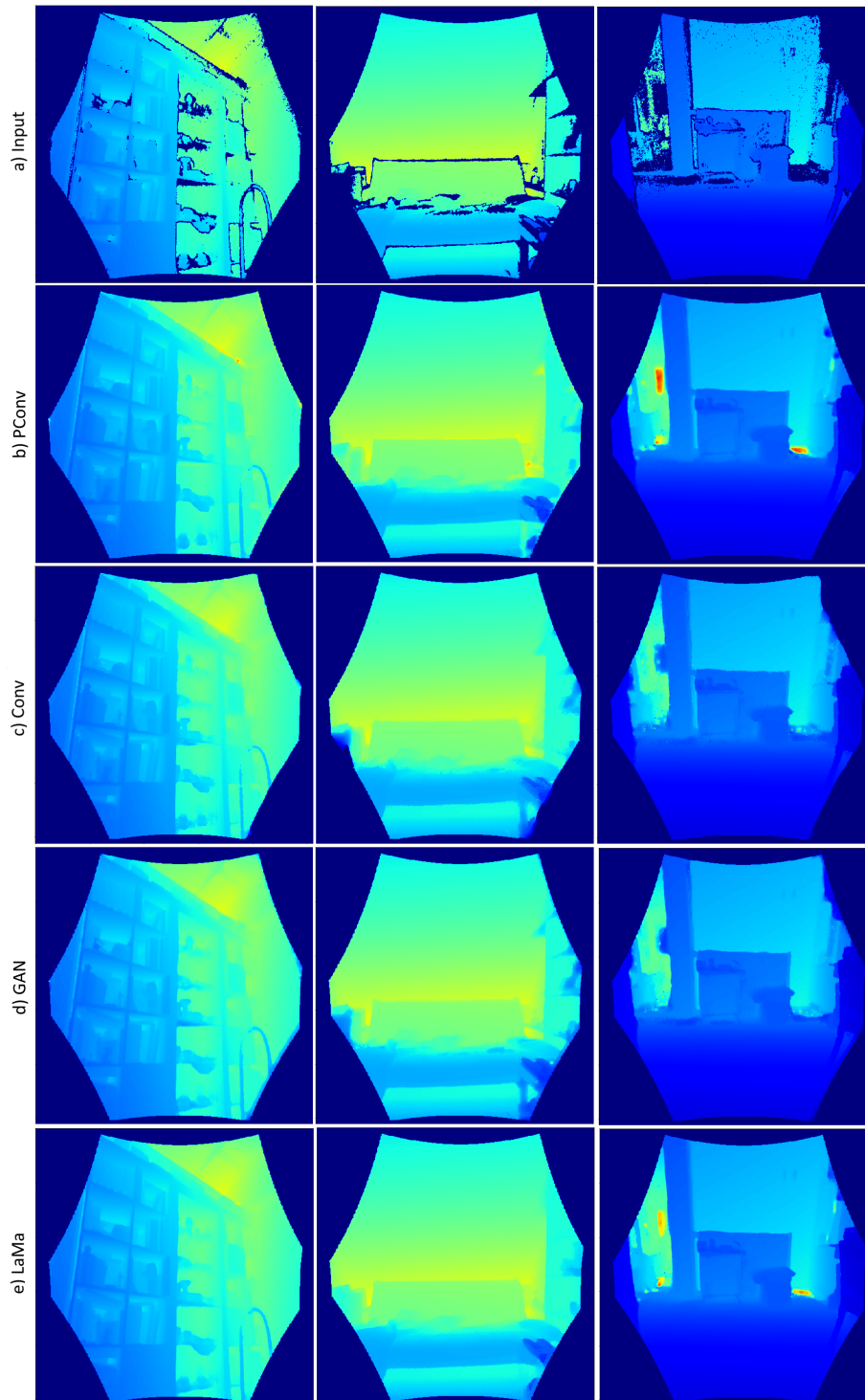


FIGURE 3.49: More inpainting results with our own dataset. Again, LaMa most often produces the best visual results. PConv behaves quite similar, and both struggles with outliers. However, in some cases, the GAN performs the best.

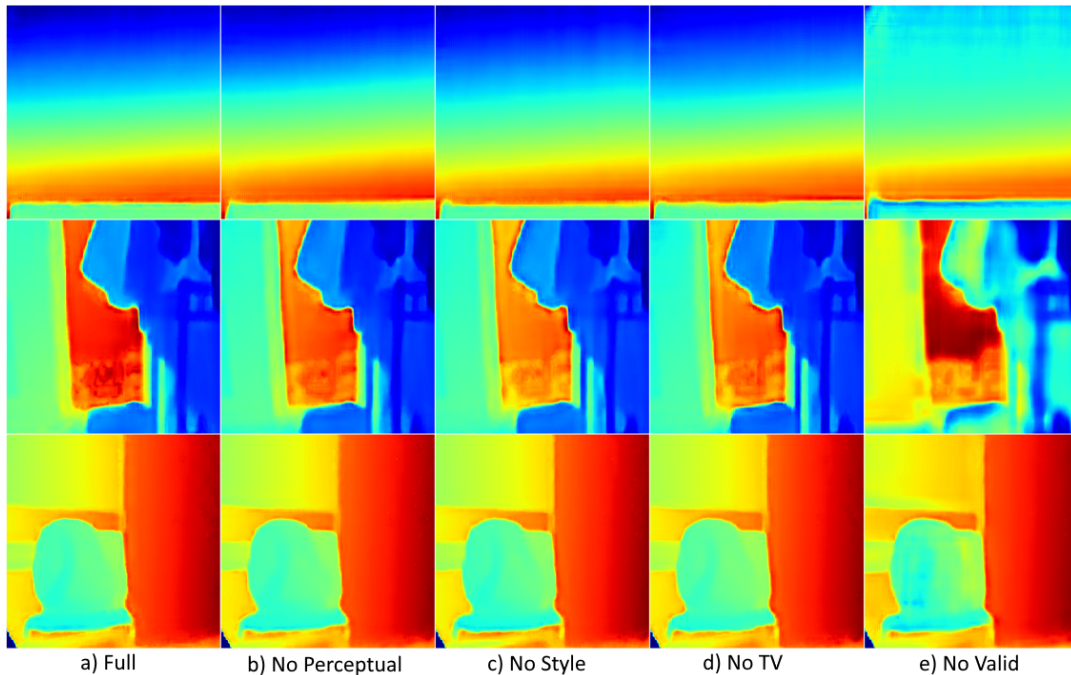


FIGURE 3.50: Ablation study on inpainted images using PConv and our dataset. Switching off any loss terms slightly degrades the results.

results, we see comparatively small changes when a loss term is switched off, see Fig. 3.50. In our case, we see the biggest impact (and degradation) when removing the valid loss. However, turning off the other loss terms also has slight negative effects on the results. Hence, the best results can be achieved when incorporating all loss terms.

### 3.3.6 Conclusions and Future Work

We presented an approach of real-time reconstruction of missing or invalid areas in depth images using deep neural networks. In particular, our approach does not use any guidance by color images. In our approach, we adopted two different U-Net-based models that originally were designed for color-image inpainting, one using partial convolutions, and the other one being a patch-based GAN. For comparison, we took also a basic U-Net and a more sophisticated state-of-the-art model, namely LaMa. The training was done using the public NYU Depth V2 dataset that we augmented with custom holes. Our quantitative and qualitative evaluations with the NYUV2 and SceneNet datasets showed that LaMa, overall, produces the best inpainting results, the GAN method performs especially well on images with smaller hole-to-image ratios, the partial convolution approach achieves consistently good results (images with various hole sizes and ratios), and the regular convolution-based approach fares the worst. Applied to a custom dataset we recorded with an Azure Kinect sensor, we found that the LaMa model, on average, leads to the visually most pleasing inpainting results, although the PConv and GAN methods also achieve reasonably good and coherent results (the latter sometimes even being superior). To conclude, all methods are able to reconstruct holes of any shape, size, or location without any postprocessing procedures, with reasonable to good visual quality. Also, except for LaMa which is notably slower, they achieve this in a real-time fashion.

---

For future work, we would find it worthwhile to also incorporate RGB data as additional input, if available, to enhance the inpainting results with this extra information. It would also be advantageous to create a more realistic noise model that accurately describes the hole occurrence in the depth images. Other network architectures such as transformer models, originally from the natural language processing domain, should be investigated to also take advantage of temporal coherency between subsequent images. Moreover, producing ground truth data for our own dataset (recorded with the Azure Kinect) would be highly beneficial for the training and evaluation of the models. One approach for this challenging task would be to couple the Azure Kinect with another precisely, externally registered depth-sensing device, such as a stereo camera, from which the depth for the missing areas can be produced.



## Chapter 4

# Perception of Teleport Visualizations in Multi User VR

After considering streaming architectures and avatar reconstruction and rendering methods for telepresence, we now shift the focus to another important aspect of multi-user VR and telepresence applications: How the participants move throughout the virtual environments. One of the most popular VR locomotion methods is the standard teleport metaphor, as it is quick, easy to use and implement, and safe regarding cybersickness. Hence, we used it, too, for our previously presented telepresence application. However, it can be very confusing to onlookers and reduce their perceived immersion and presence. The reason for this is the discontinuity of the process, and, therefore, the lack of motion cues. We briefly considered this issue in Section 1.1.

Now, in this chapter, we deal with the question of how this popular teleport metaphor can be suitably visualized to onlookers in multi-user VR environments. To find an answer to this question, we implemented several continuous and discontinuous 3D visualizations for the teleport metaphor and conducted a user study for evaluation. Specifically, we investigated them regarding their ability to prevent confusion, and spatial awareness as well as perceived spatial- and social presence.

The work presented in this chapter is based on our submitted paper PP1 in Appendix A.

### 4.1 Introduction

In the previous chapters, we learned that continuous technological advances and decreasing costs lead to a growing popularity of virtual reality among researchers, developers, and consumers alike. The ability to immersively experience virtual environments as if actually present makes VR highly interesting for applications ranging from gaming and entertainment to training and education [414]. See also Section 2.3 for a wide range of example applications. Furthermore, multi-user VR applications provide co-located or remote participants with the ability to freely interact with each other and collaborate in shared virtual environments, which has been shown to be highly beneficial for a wide array of tasks such as liver surgery planning [62], moderated remote usability testing [54], and computer-aided design and construction [387]. In these multi-user VR environments, the users get virtually represented by 3D avatars to enable effective interaction and collaboration. Especially full-body avatars that realistically depict the user have been shown to be advantageous for the sense of presence, embodiment [356], trust formation, and task performance [266]. More information about avatars can be found in Section 2.2.2.

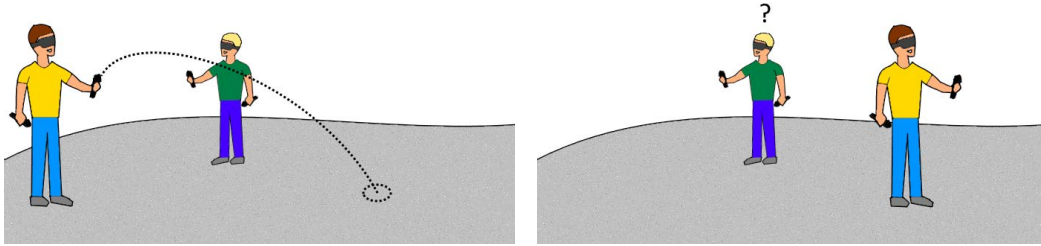


FIGURE 4.1: Depiction of the possibly confusing teleport locomotion in multi-user VR. Left: User teleports to a new location. Right: After the instantaneous, discontinuous teleport, the observing user lost track and is confused.

One of the most important design decisions for a VR system is which type of locomotion metaphor should be used for traveling greater distances in the virtual environment [295]. As a reminder, we discussed this and adjacent topics in Section 2.2.1. To reiterate the key points, smaller movements are usually handled well using “room-scale”, in which the actual movements get tracked and directly replicated. Typical locomotion methods for greater distances include (point&click) teleportation, redirected walking, walk-in-place, and steering (e.g., via joystick). All these techniques naturally have different strengths and weaknesses regarding aspects such as physical effort, precision, time, etc. [4]. The teleportation metaphor, for instance, is among the most popular ones, as it is relatively simple, quick, and proven to be unlikely to induce cybersickness [30, 42]. Cybersickness is a major concern for VR applications. It is similar to motion sickness and is believed to be mainly caused by a sensory mismatch between the visual and the vestibular and proprioceptive systems [204].

Using the standard teleport locomotion metaphor in multi-user environments, however, has one significant drawback: The inherent discontinuity of the process may disrupt multiplayer gameplay and lead to confusion for observers when the user(s avatar) seemingly vanishes or emerges from nowhere, as argued by Griffin and Folmer [133] and reported incidentally by Wang et al. [394]. Figure 4.1 depicts this scenario. Moreover, this behavior could be easily mistaken to be the result of network issues, in fact, it would strongly resemble a high-lag connection in online gaming. Ultimately, the chance for a loss of presence for the observers would be, presumably, significant. Especially so as it was already established that the abrupt change of location and the absence of any motion cues can lead to confusion and a loss of presence for the teleporting user himself [2]. Presence – the sense of being there/in the virtual environment – is a crucial factor for the quality of the VR experience, though [330, 300], and should be as high as possible. More about the concept of presence can be found in Section 2.1.2. Prithul et al. [287], too, identified the issue of teleportation in multiplayer scenarios and anticipate detrimental effects to the presence of observers. Hence, they view it as an important direction for future research.

With this work, we want to tackle this issue and expand on the very sparse research on this topic. Typical dictionary definitions for something “confusing” are: “Something that is confusing makes it difficult for people to know exactly what is happening or what to do” [87], or “... because it is difficult to understand” [86]. Therefore, we have implemented several visualization methods to convey the deliberate act of teleportation to observing users in a shared virtual environment, similar



to Thanyadit et al. [377] and Freiwald et al. [113]. Our main goal is to do a comprehensive evaluation of the effects of different teleport visualizations on observers. Specifically, if the visualizations enhance spatial awareness, prevent confusion, and, thus, a loss of presence. Therefore, we have formulated the following research questions:

- $R_1$ : Do visualizations help with preventing confusion caused by the teleportation process?
- $R_2$ : Do visualizations help to retain presence when teleporting?
- $R_3$ : Do continuous visualizations provide more spatial awareness?
- $R_4$ : Which visualization is generally the best (e.g., presence, confusion, user preference)?
- $R_5$ : Do the visualizations differ regarding the distance scalability?

To answer these questions, we have conducted a subsequent study, in which we investigated the visualizations' effects on observers, i.e., regarding spatial awareness. In order to guarantee similar conditions and minimize confounding effects, we opted for a study design with a single observer that views pre-recorded teleportations. In contrast to previous work, we tested multiple scenarios (in/out of-FoV) and properties such as the spatial and social presence, the plausibility and intuitiveness of the visualization as well as the visualizations' distance scalability. With our user study and extensive evaluation, we provide valuable insights into this crucial but under-investigated topic. Moreover, in our study, we did also investigate the influence of the visualization duration/speed.

## 4.2 Related Work

Locomotion in virtual reality was and still is extensively researched. The previous works were most commonly concerned with designing novel advantageous locomotion metaphors, as well as examining and comparing them regarding factors of interest such as cybersickness, presence, user preference, and effectiveness. For instance, Mine [252] was one of the first who described teleportation as a locomotion metaphor, Bozgeyikli et al. [36] further evaluated the point&teleport locomotion, Weller et al. [406] recently proposed a novel approach to redirected walking, Wilson et al. [413] introduced a new method for the walking in place metaphor, Zeleznik et al. [434] adapted the classic teleport to be a quick dash that provides some optical flow/motion cues and Bhandari et al. [24] compared this technique to regular teleportation. Interestingly, the latter found no heightened cybersickness using the dash instead of the standard teleport. A much more comprehensive overview of the different locomotion metaphors is given by Boletsis [30]. Other works such as the one by Christou and Aristidou [67] focused on deeper analysis and comparisons of the locomotion metaphors. In this case, the authors compared steering with teleportation and found the former to lead to more cybersickness and the latter to be faster yet equally effective. Mayor et al. [241] also analyzed different locomotion methods regarding presence, cybersickness, and usability and came to similar results. Boletsis and Cedergren [31] examined a selection of prevalent locomotion techniques regarding their user experience, Lesaca et al. [209] compared different teleport implementations, and Kruse et al. [191] investigated several jumping gestures for teleportation. Moreover, Kelly et al. [181] found that rotational self-motion cues positively affect

the spatial updating performance when teleporting. More information specifically about teleport locomotion can be found in the work by Prithul et al. [287]; generally, it is found to be highly performant and safe regarding cybersickness, but also prone to spatial disorientation in the absence of any motion cues and only limited path integration. Adhikari et al. [2] proposed a hybrid solution between continuous and teleport locomotion to merge their respective benefits by adding a series of smaller teleport actions to continuous leaning/steering locomotion. With a similar goal, Griffin and Folmer [133] developed the out-of-body locomotion technique that switches to a third-person view when a teleport is executed and in which the avatar gets continuously steered. Similarly, [71] lets the user switch to a bird's eye view and navigate from there using raycast aiming.

All these prior works focus on the locomotion method's effects on the navigating user himself, though, and do not consider the observer's perception in multi-user environments. Even when looking at the related research topic of group navigation, where the focus lies on designing systems in which groups of people can navigate a common virtual space together, the findings about multi-user locomotion visualization are sparse. In this area, the main goals are finding and maintaining suitable formations, as well as, object avoidance, and improving the comprehensibility of the process. Recent examples from this area are the works by Chheang et al. [64] and Weissker and Froehlich [403].

How locomotion by teleportation is perceived by others and how it could be favorably visualized is hardly researched yet. Accordingly, Prithul et al. [287] came to the same conclusion in their review about the teleportation metaphor. To our knowledge, the only works that explicitly looked into this topic are the ones by Thanyadit et al. [377] and Freiwald et al. [112, 113]. The latter first compared steering-based locomotion with teleportation, as well as avatar appearance, in a competitive multi-user, shared virtual environment. The task was to play a virtual match of snowball and hit the opponent as often as possible. While the player who was to be questioned was limited to the steering locomotion, the opponent either used steering too or teleportation. The authors found that the continuous steering locomotion ranked significantly higher regarding co-presence and perceived fairness, while the avatar's appearance had only a negligible effect. These results reinforce the assumption of a reduction in the presence and other adverse effects of teleportation in contrast to continuous locomotion methods. In the subsequent work, Freiwald et al. [113] did focus on the issue of discontinuous teleport locomotion in shared VR environments. To create a better experience and increase spatial awareness for observers, they proposed a system that temporally depicts special "smart avatars" that mimic the locomotion of the user to the observers. The idea is that these avatars do a continuous transition to the target destination, although the actual teleport is discontinuous. Four different transition techniques were implemented and evaluated regarding spatial awareness, attractiveness, and pragmatic and hedonic quality scores. Generally, the transitions consist of a walking animation, or depict some kind of trail. The proposed continuous transitions were rated higher for all these factors.

Thanyadit et al. [377], which, together with the later work by Freiwald et al. [112, 113], is the work most closely related to ours, did also identify the unique issues of teleportation in multi-user settings. Hence, they designed 4 substituted visualizations as a remedy, namely: hover, jump, fade, and portal. The former two are rather similar and resemble the continuous dash locomotion technique. The fade visualization slowly fades the avatar out and in, at the start and target location, respectively. The portal method uses separate portals to achieve a similar effect. The authors also

identified general design requirements, being time efficiency, traceability, intuitiveness, and recognizability, and briefly discussed the visualizations. Moreover, they did a pilot study with 5 participants that found the hover visualization to be the preferred one.

Although their work featured multiple similar visualizations as well as evaluation properties as ours, they lack a large, formal evaluation. To this date, the only comprehensive evaluation of teleportation visualizations for observers is the one by Freiwald et al. [113]. However, in contrast to us, they did not test multiple scenarios, e.g., the influence of teleportation strictly in the observer's FoV compared to teleporting in/out of it, or the visualizations' distance scalability. Moreover, the visualizations' effect on the observers' spatial- and social presence as well as how plausible and intuitively understandable the visualization depicts the teleportation process were not evaluated, yet.

### 4.3 Proposed Teleport Visualizations

In order to investigate which visualization would be best suited to convey the teleportation process to observers in a multi-user setting, we decided to implement a variety of visualization methods. Important properties which we took into consideration were the degree of predictability and traceability the visualization provides, the time a convincing representation would take, the intuitiveness of the visualization, and the general plausibility. The first three properties are similar to the requirements Thanyadit et al. [377] proposed. The plausibility, naturally, is dependent on the exact setting and the user's representation itself, i.e., the kind of avatar. For our investigation, we focused on full-body avatars and a generic environment setting which should make the results more widely applicable. In addition to the standard point&click teleport that instantly changes the avatar's location, we opted to implement a teleport with particle trace, a portal metaphor, a beam particle effect, a quick dash, and a complete walking animation. As all these other methods take time, in contrast to the standard teleport, we also included a delayed teleport for a time-normalized comparison.

In the following, we briefly describe the individual visualizations.

- The standard point&click teleport (P&C, or just teleport from now on) instantly changes the character's location to the target destination without any visual feedback to observers, see Fig. 4.2 (a). Although the teleport line/arc that the user himself usually can see could also be visualized for observers, we decided against it as this is not the usual practice. Additionally, we have a delayed implementation in which the character only arrives after the same amount of time as with the other visualizations (P&C S, or slow teleport).
- The particle trace (P. Trace) visualization is implemented using a particle system and shows many continuously emerging (and slowly fading out) particle spheres along the path from start to destination, see Fig. 4.2 (b). This trace of particles provides a motion cue to observers. This metaphor is inspired by several computer games using similar techniques, such as League of Legends ("Pike" character), as well as, the "dissolve" transition by Freiwald et al. [113].
- The beam effect is a warping- and glowing effect briefly applied to the character's material at the start and end of the teleportation, see Fig. 4.2 (c). It resembles beam effects in many sci-fi movies (e.g., Star Trek) and computer games

TABLE 4.1: Properties of the visualizations. Continuous ones should be more traceable but less time efficient. Walking should be intuitive and plausible, the teleport not.

Property	P&C	P. Trace	Beam	Portal	Dash	Walking
Traceability	-	+	-	0	+	+
Time Efficiency	+	-	+	0	-	-
Intuitiveness	-	0/+	+	+	0	+
Plausibility	-	0	0	0	0	+

and is also part of the “dissolve” transition by Freiwald et al. [113] and similar to the “Fade” visualization by Thanyadit et al. [377]. It is non-continuous and rather quick.

- The portal metaphor is generally similar to the beam effect, see Fig. 4.2 (d). However, instead of applying a visual effect on the character, a portal emerges through which he then steps. At the destination, another portal pops up from which the character reemerges. It is similar to the one by Thanyadit et al. [377].
- The dash visualization is similar to the one in [24] and the hovering metaphor in [377], see Fig. 4.2 (e). The full-body avatar is quickly and continuously translated in a direct line to the destination, thus, providing the observer with motion cues, similar to the particle trace.
- The walking visualization is a fully-fledged, pre-recorded walking animation that is played and shown to the observers, see Fig. 4.2 (f). This should help to increase the plausibility and be the most natural visualization, depending on the speed/distance.

Table 3.1 shows an overview of the visualizations we decided to implement and a comparison of their properties. Again, the plausibility is highly context-dependent.

In general, the visualizations can be classified as continuous and non-continuous ones. The former usually feature higher tractability and presence but tend to be less time efficient (at least for the teleporting user) [70]. Example visualizations for this category would be the dash, the particle trace, and, on the far end of the spectrum, the full walking animation. Non-continuous visualizations would be the beam effect, the portal, and, on the other end of the spectrum, the extreme case of the standard teleport.

## 4.4 Study

To conduct a user study about the teleportation depiction and its effects (see the last paragraph of Section 4.2), we have implemented all the aforementioned visualizations using the Unreal Engine 4.26. In the remainder of this section, we first list the hypotheses that we formulated based on our research questions, which we stated at the end of Section 4.1, then describe the experiment which we designed for this study, and finally, also detail the experiment’s procedure.

### 4.4.1 Hypotheses

Based on prior work about locomotion in virtual reality, and our own considerations, we defined the following eight hypotheses to answer our research questions.

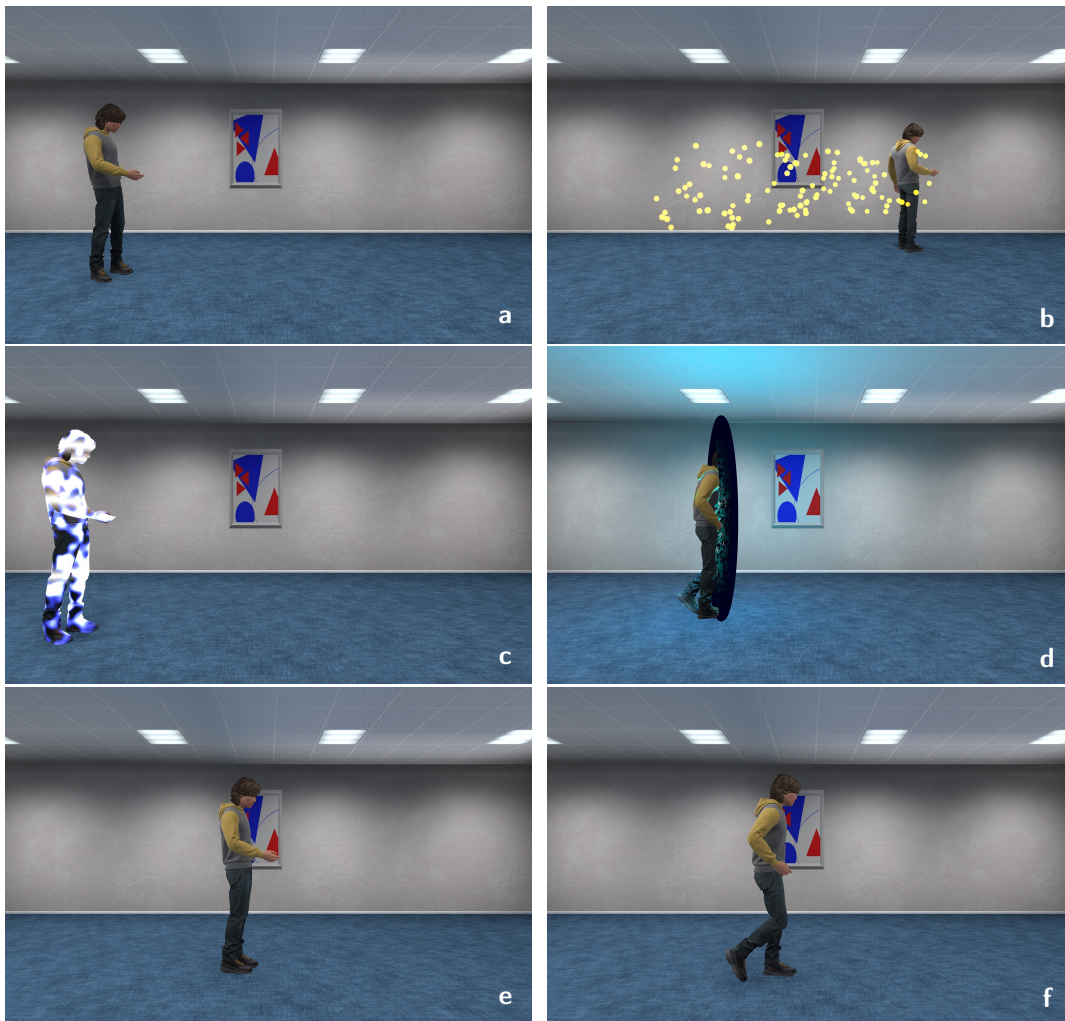


FIGURE 4.2: Our teleport visualizations: None (the standard point & click teleport and the delayed variant) (a), particle trace (b), beam (c), portal (d), dash (e), and walking (f).

From research question  $R_1$ , and the report by Wang et al. [394], and the definition of confusing (see Sec. 4.1) we directly derive hypotheses  $H_1$  and  $H_2$ :

- $H_1$ : A teleport visualization makes the locomotion process more intuitively understandable.
- $H_2$ : A teleport visualization makes the locomotion process more plausible.

Similarly, to answer research question  $R_2$ , we formulate hypotheses  $H_3$  and  $H_4$ :

- $H_3$ : A teleport visualization has a positive impact on the perceived spatial presence.
- $H_4$ : A teleport visualization has a positive impact on the perceived social presence.

It was already established that continuous locomotion tends to provide a higher presence for the teleporting user [70], thus, it arguably holds also for the observers.

To check research question  $R_3$ , we decided to focus on the abilities to track and relocate a person, thus, we raise the hypotheses that

- $H_5$ : Continuous teleportation visualizations increase the ability to track the person.
- $H_6$ : Continuous teleportation visualizations increase the ability to quickly relocate the person.

This is a natural assumption to make, as a continuous visualization directly provides visual cues to the observer. Moreover, Freiwald et al. [113] reported higher spatial awareness for observers by using continuous visualizations for teleport locomotion.

As the walking animation is the most natural metaphor (at least when the speed is appropriate), we answer the research question  $R_4$  by hypothesizing (similar to Freiwald et al. [113]) that

- $H_7$ : The walking animation is preferred the most by the users.

Lastly, we would expect that the continuous visualizations have inherently a more limited range of distances/speeds in which they are convincing and effective. Therefore, to answer research question  $R_5$ , we formulate the hypothesis  $H_8$ :

- $H_8$ : Continuous teleport visualizations exhibit a lower distance scalability.

#### 4.4.2 Experimental Setup

To test the teleport visualizations, we created a 3D office scene in the Unreal Engine 4.26. As to not distract the participants, the 3D scene is rather minimalistic and free of clutter, yet, the lighting and the used meshes are of high quality. We implemented all the teleport visualizations that we described in Chapter 4.3 and opted to use a high-fidelity MetaHuman avatar for the teleporting character. However, to keep the performance reasonably high, we had to lower the avatar's hair's fidelity. The participants were supposed to stand in the middle of the room and were not represented by any avatar themselves, again, to minimize distractions. To guarantee comparable conditions for all participants and to minimize confounding effects between subjects and between different visualizations, we pre-recorded animations using the OptiTrack motion capture system for the teleporting character to perform

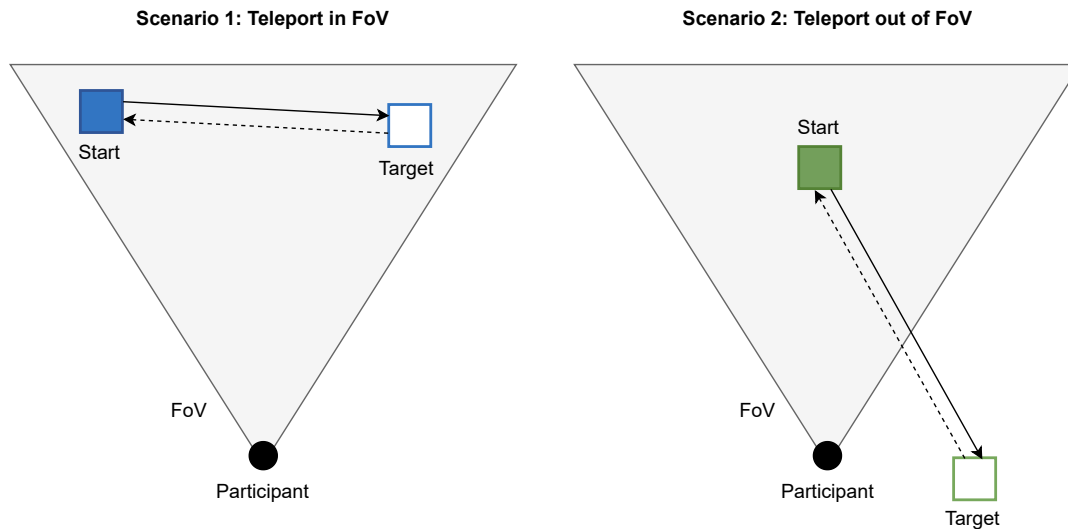


FIGURE 4.3: Top-down view of the scene setup of our experiment in VR. Scenario one (in-FoV) is depicted in blue (left), and scenario 2 (out-of-FoV) is in green. Both scenarios consist of a two-way teleportation to the destination (full arrow) and back (dotted arrow). The exact positions and angles were randomized.

and opted to only have one observer at a time. For each of the different teleport visualizations, the procedure was the same: the character looks around briefly, executes the teleport by pointing the controller in the destination direction, the visualization is shown, and the character arrives at the target destination. Finally, the character turns around and teleports back to the original position. We chose this two-way teleportation path in an effort to maximize the effect size.

In our experiment, we opted to test two scenarios in a within-subject design. In the first one, the character starts in full view of the participant and teleports either from left to right or the other way around. The destination (and the full path) was always in view, too. We chose this teleport setup in order to have distances as large as possible in the FoV. The order in which the teleport visualizations were applied throughout the use case was randomized. Also, to reduce the predictability, possible mental fatigue and prevent confounding learning effects, the exact teleport angle, and therefore target destination, was slightly randomized. The distance was always the same, though. For the second scenario, we focused on a more advanced setting with a higher potential for a reduction in presence, namely, when the target destination is out of view of the observing user. The setup was principally the same as before, however, this time, the character's target destination was set up to be behind the observing user. Again, the visualizations' order as well as the exact target destination were slightly randomized, meaning the character sometimes teleported to the observer's back left or back right. Figure 4.3 depicts the setup for both scenarios. The participants always had to go through both scenarios, although the order was randomized.

In order to quantitatively measure the observers' ability to find or track the teleporting user, we used the HMD's built-in eye-tracking system and measured the time the participants looked (roughly) at the teleporting character. Additionally, the participants had to point and track the character with a virtual laser pointer, which

we again tracked.

Additionally, to investigate the effect of the visualizations' duration and their scalability regarding the teleportation distance, we decided to perform our experiment again with a faster teleport/visualization speed. Empirical tests we conducted led us to use the durations of 1.42 seconds for the original, slower variant and 0.71 seconds for the faster one over a teleport distance of 2.88 meters. Thus, theoretically, the user had a speed of 2.03 m/s and 4.06 m/s. The animations were sped up accordingly from their original speed to match the durations. The eventual durations were always the same for each visualization during the experiment, with the exception of the standard teleport, which is performed instantly.

Although we employ a within-subject design for the two scenarios (in/out of the field of view) to get sound results with a reasonable amount of participants, for the second, faster experiment variant, we made sure to recruit new participants that did not take part in the original one to reduce the repetitiveness. This makes our study, ultimately, a mixed design study (a combination of a between-subjects design and a within-subjects design). We find this to be a good compromise.

### 4.4.3 Procedure

The study procedure, which is depicted in Fig. 4.4, started with the participants being informed about the study and its purpose and them giving their consent. However, the exact goal was not revealed to them; only that it involved multi-user VR. Then, they were asked to fill out a pre-questionnaire with demographic data, their experience in VR and with games with avatars, cybersickness, etc. After this, the participants were given a minute to familiarize themselves with the HMD and the virtual 3D environment. Additional training was not necessary, as the task was simply observing and pointing at the avatar. Eventually, the actual experiment (either the slow or fast variation) started in which the participants had to observe the teleportation visualizations (one after another), track the character, and answer our questionnaires (see below). This procedure then was repeated directly for the other of the two scenarios. Finally, they were again asked about cybersickness and any comments on a post-experiment questionnaire.

We decided to let the participants complete the questionnaires directly in VR. The reason is that recent research on using presence questionnaires in VR suggests that this reduces the time needed for adjusting between VR and the real world, reduces potential distracting cues from the real world, and, most importantly, reduces the occurrence of breaks in presence [330]. Concretely, after each visualization throughout a scenario, the user was presented with a black screen on which text questions about spatial and social presence appeared one after each other. At the same time, the questions were also read out loud by an assistant. The verbally given answers were written down by the assistant. We decided to limit ourselves to two questions each to reduce the repetitiveness and the time. The exact questions are listed in Table 4.2 and had to be answered using a 7-point Likert scale. The questions are a subset we carefully selected from the more comprehensive presence questionnaire by Makransky et al. [231]. In addition to answering the questions about presence, the participants had to rank the visualizations regarding several criteria, such as plausibility or target anticipation, after completing each of the two scenarios. The criteria are listed in Table 4.3. This ranking was also performed in VR. For the ranking, the participants were presented with a gallery of enumerated images of all the visualizations, serving as a reminder, and had to order them, one time per criteria. Again, the verbally given answers were written down by the assistant.



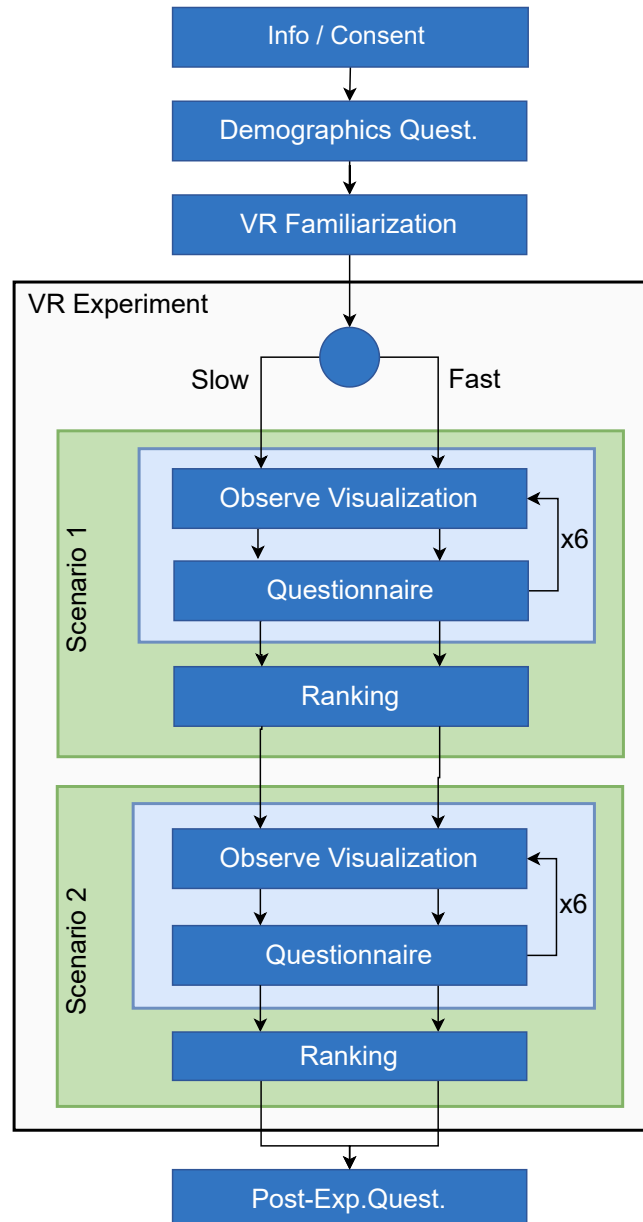


FIGURE 4.4: Diagram of the study procedure. Each participant experienced both scenarios (in-FoV/out-of-FoV) but took part only in either the slow or the fast experiment variant.

TABLE 4.2: Our questionnaire about social presence (1./2.) and spatial presence (3./4.), which is based on the Multimodal Presence Scale [231]. Answers from “None” to “Very much” using a 7-point Likert scale.

- 
1. I felt like I was in the presence of another person in the virtual environment.
  2. The person in the virtual environment appeared to be sentient (conscious and alive) to me.
  3. The virtual environment seemed real to me.
  4. While I was in the virtual environment, I had a sense of “being there”.
-

TABLE 4.3: Our questionnaire in which the participants had to rank the visualizations according to the various criteria listed below. Each item begins with “Order the visualizations by”.

- 
1. how plausibly they represented the movement.
  2. how intuitively they represented the movement.
  3. how well you could anticipate the destination of the other person with the help given by the visualizations.
  4. how fast you could find the other person after the locomotion took place.
  5. perceived speed.
  6. how much you liked them.
- 

## 4.5 Results

Here, we first describe the participants and demographic data and then present qualitative data from our questionnaires as well as additional quantitative results.

### 4.5.1 Participants

For our study, we recruited  $n = 52$  participants with a distribution of 76.9 % men and 23.1 % women. In our mixed design, a random half of the participants took part in the slow experiment variant and the other half in the fast one, but all participants experienced both scenarios. The participants’ ages’ ranged between 18 and 71 years with an average age of 30.98 (SD = 12.9) years. Asked about previous experience with VR, 36.5 % reported to have none or very little (less than 5 times), 17.3 % stated to have moderate experience (5 to 10 times), and 46.2 % had extensive experience (more than 10 times). Furthermore, 15.4 % of the participants stated to not have any awareness of the teleportation metaphor for locomotion (in general, not necessarily regarding VR), while 34.6 % reported to be not familiar with seeing another player as an avatar in virtual 3D worlds. Regarding previous experience with multiplayer games (first/third-person only), 23.1 % reported to have none or very little (less than 5 times), 17.3 % stated to have moderate experience (5 to 10 times), and 59.6 % had extensive experience (more than 10 times).

### 4.5.2 Qualitative and Quantitative Data

For our presence data, which we gathered using questionnaires with 7-point Likert scales, we assumed the data to be normally distributed. This assumption was confirmed by Shapiro-Wilk tests that we performed for validation. To statistically evaluate our data, we then conducted repeated measure ANOVA to check for statistically significant differences between groups, followed up by pairwise posthoc testing using dependent samples t-tests with Bonferroni correction, to find the exact groups with significant differences. As we did employ a different measurement method for our second questionnaire, namely relative rankings between the visualizations, we did assume to have not normally distributed data. This was confirmed by Shapiro Wilk-tests. Therefore, we employed the Friedman test, followed up by a pairwise Bonferroni corrected Wilcoxon signed-rank test. Lastly, for the tracked hit data, we had no definitive assumption for the distribution, which is why we again performed Shapiro-Wilk tests. As they were not normally distributed according to

the tests, we employed the non-parametric evaluation process for this data. We always assumed the level of significance (alpha) to be 0.05, as usually done. Note that we directly show only one plot of the four scenario-variant combinations for each investigated criterion to keep the evaluation reasonably long and clear. We selected the ones that we found most representative. For the same reasons, we describe only the statistical details for the slow experiment variant in detail, as we find it more meaningful. However, in the discussion part, we also consider the most interesting results of the fast variant and all the statistical data is listed in corresponding tables at the end of the result section. Moreover, all plots can be found in Appendix B.

**Cybersickness** In regard to cybersickness, the average Likert Score before the experiment was 1.44 ( $SD = 0.84$ ), which increased slightly to 1.59 ( $SD = 1.09$ ) after the experiment. Specifically, before the experiment, 75.0 % of the participants reported not having any cybersickness (no feeling of nausea, dizziness, or discomfort). 21.1 % of the participants (11 participants) reported having just slight feelings of nausea, dizziness, or discomfort, and 3.8 % (2 participants) reported moderate levels of such feelings. The cybersickness ratings after the experiment stayed mostly the same, or increased marginally, for the participants who had originally none. Only one participant started to have a strong feeling of cybersickness. The two participants that originally reported moderate cybersickness, reported after the experiment to have less. Participants that originally reported slight feelings of nausea, dizziness, or discomfort, reported mostly the same afterward.

**Presence** To measure the presence, we aggregated the two social presence and spatial presence questions, respectively. For the social presence in the “in the field of view” (IFoV) scenario and slow variant, the walking visualization got the highest ratings ( $M = 4.42, Mdn. = 4.75, SD = 1.87$ ), while the slow teleport ( $M = 3.25, Mdn. = 3.5, SD = 1.45$ ) and dash ( $M = 3.25, Mdn. = 3.5, SD = 1.48$ ) got the lowest. We found the data to be normally distributed and ANOVA ( $p = 0.0001$ ) revealed significant differences between the visualizations. Posthoc testing revealed these to be between the slow teleport and walking ( $p = 0.0038$ ), teleport and walking ( $p = 0.0445$ ), and dash and walking ( $p = 0.018$ ). Note, in our notation, the latter visualization is always the one that was rated higher. For the “out of the field of view” (OFoV) scenario, we got similar results: the walking visualization was again rated highest ( $M = 4.31, Mdn. = 4.5, SD = 1.73$ ), the dash ( $M = 2.98, Mdn. = 2.5, SD = 1.5$ ) and slow teleport ( $M = 3.37, Mdn. = 3.25, SD = 1.41$ ) the lowest, see Fig. 4.5 (top). Again, the data was found to be normally distributed and ANOVA revealed significant differences ( $p < 0.0001$ ). Posthoc testing showed significant differences between the slow teleport and walking ( $p = 0.0024$ ) and dash and particle trace/beam/portal/walking ( $p = 0.0437/0.0326/0.0477/0.003$ ). The pair of teleport and walking barely missed the threshold with a  $p$  value of 0.0521.

For the spatial presence, we found the ratings generally to be slightly higher. In the IFoV scenario, the walking visualization was again rated the highest ( $M = 5.19, Mdn. = 5.5, SD = 1.32$ ), the rest were closer together this time, and the portal was rated lowest ( $M = 4.35, Mdn. = 4.0, SD = 1.43$ ). As before, ANOVA indicated significant differences ( $p = 0.0007$ ), and posthoc testing revealed them to be only between the dash and the walk visualization ( $p = 0.0105$ ). However, the pair of slow teleport and walking missed the threshold just slightly with  $p = 0.0683$ . In the OFoV scenario, walking was rated highest again ( $M = 5.23, Mdn. = 5.25, SD = 1.3$ ), but this time the dash was rated lowest ( $M = 4.23, Mdn. = 4.25, SD = 1.58$ ), see Fig. 4.5

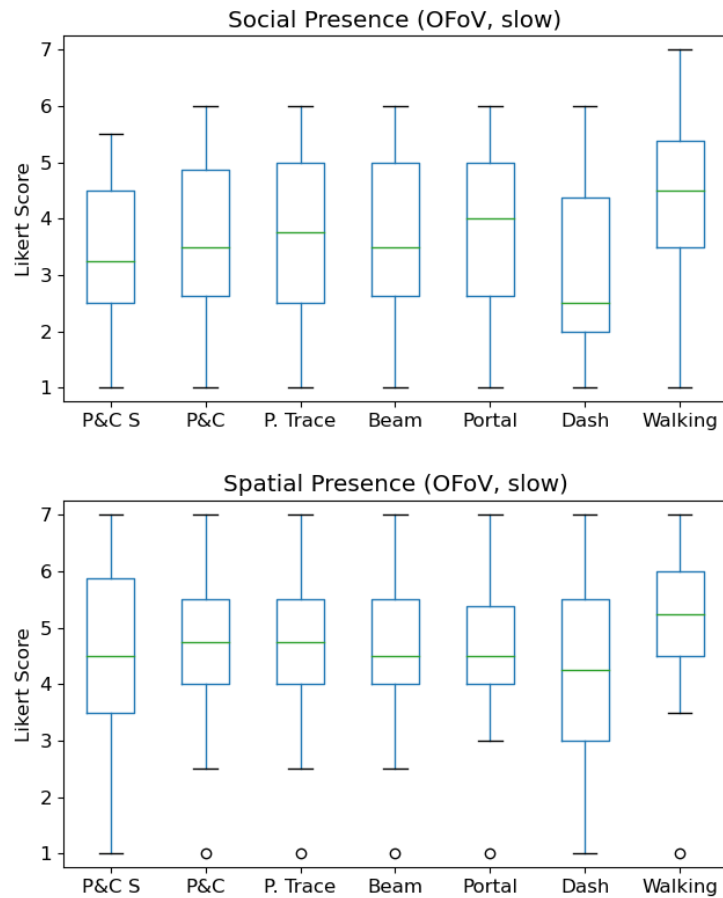


FIGURE 4.5: Social (top) and spatial (bottom) presence results in the out-of-FoV scenario and slow variant. The walking visualization is rated highest while the dash is rated lowest.

(bottom). ANOVA returned  $p < 0.0001$  and posthoc testing revealed significant differences between the slow teleport/particle trace/beam/portal/dash and walking ( $p = 0.0273/0.0202/0.0358/0.0184/0.01$ ). The pair of teleport and walking missed slightly with  $p = 0.0708$ .

**Plausibility and Intuitiveness** Regarding the questions of how plausible and intuitively understandable the visualizations were, we got rather similar results, also for both scenarios (IFoV/OFoV). In all cases, walking was rated the highest (i.e., plausibility, IFoV:  $M = 5.42$ ,  $Mdn. = 6$ ,  $SD = 1.27$ ), and both teleports (i.e., plausibility, IFoV:  $M = 1.77/2.54$ ,  $Mdn. = 2/2.5$ ,  $SD = 1.42/1.78$ ), as well as the dash (i.e., plausibility, IFoV:  $M = 2.11$ ,  $Mdn. = 1.5$ ,  $SD = 2.08$ ), were rated the lowest, see for example Fig. 4.6. The data was found to be not normally distributed. After the Friedman test, which indicated significant differences, we found them in posthoc testing to be between the walking visualization (rated higher) and all other ones regarding plausibility, for both the IFoV scenario ( $p = 0.0002/0.001/0.0061/0.006/0.0169/0.0001$ ), and the OFoV scenario ( $p = 0.0002/0.0003/0.0191/0.0143/0.0119/0.0002$ ). In the case of intuitiveness, significant differences were found again for the walking visualization (rated higher) and all other ones (except the particle trace in the OFoV scenario). In the IFoV scenario, we got:  $p = 0.0002/0.0003/0.0184/0.02/0.0003$ ,

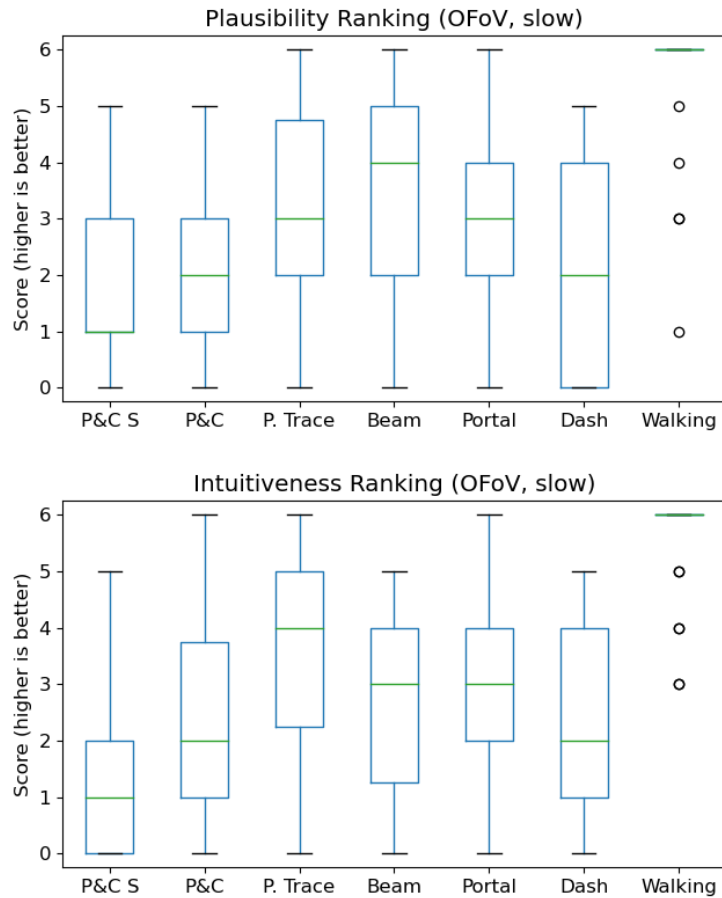


FIGURE 4.6: Results for the plausibility (top) and intuitiveness (bottom) rankings in the out-of-FoV scenario and slow variant. Walking is rated highest, while both teleport variants and dash got rated the lowest.

and in the OFoV scenario  $p = 0.0002/0.0004/0.0004/0.0065/0.0004$ . Moreover, in the IFoV scenario, both teleports were rated significantly lower than the beam ( $p = 0.0028/0.0374$ ), and in both scenarios, the slow teleport was rated lower than the particle trace (IFoV:  $p = 0.0028$ , OFoV:  $P = 0.0436$ ).

**Target Anticipation and (Re-)Spotability** Our results regarding target anticipation as well as the ease of (re-)spotting the person after the locomotion show that the walking visualization was rated the highest for both criteria and both scenarios (i.e., target anticipation/spotability, IFoV:  $M = 5.15/5.46$ ,  $Mdn. = 6$ ,  $SD = 1.12/1.07$ ), see for example Fig. 4.7. The dash and particle trace followed suit, while both teleports (i.e., IFoV, target anticipation:  $M = 1.08/1.5$ ,  $Mdn. = 1/1$ ,  $SD = 1.2/1.6$ , i.e., IFoV, spotability:  $M = 0.46/1.15$ ,  $Mdn. = 0/1$ ,  $SD = 0.65/1.12$ ) were rated lowest. The data was again not normally distributed. Significant differences were found between many visualizations: In the case of target anticipation, we found significant differences between the beam visualization and particle trace/dash/walking (IFoV:  $p = 0.0014/0.0138/0.0003$ , OFoV:  $p = 0.0007/0.0046/0.0001$ ), between the slow teleport and particle trace/portal/dash/walking (IFoV:  $p = 0.0003/0.0387/0.0017/$

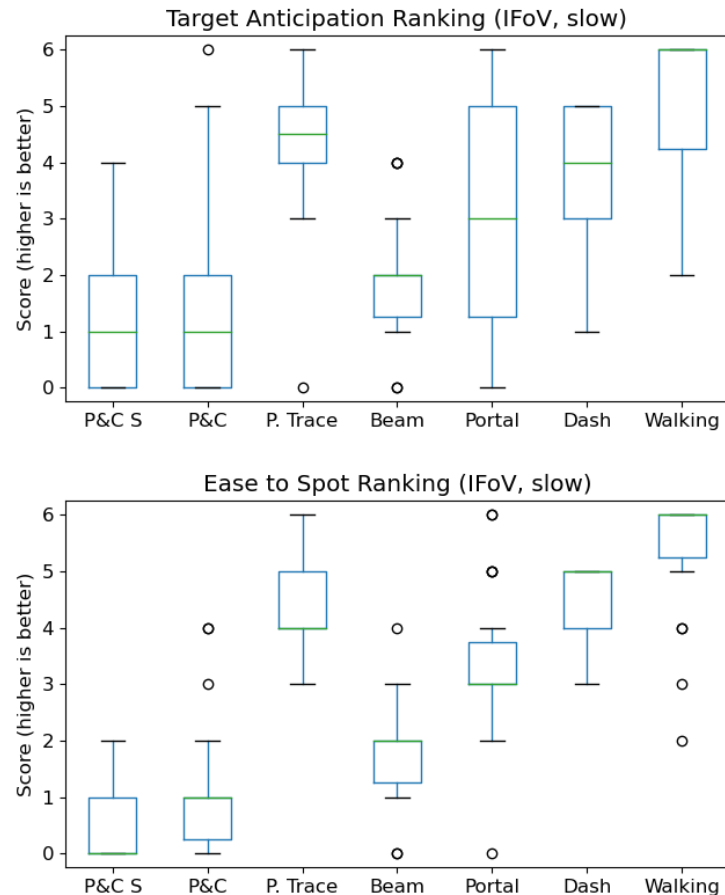


FIGURE 4.7: Results of the target anticipation (top) and (re-) spotability (bottom) rankings in the in-FoV scenario and slow variant. Walking is rated the highest, particle trace and dash follow suit. Both teleports rank the lowest.

0.0002, OFoV:  $p = 0.0008/0.0034/0.0002/0.0002$ ), between teleport/dash and walking (IFoV:  $p = 0.0002/0.0007$ , OFoV:  $p = 0.0003/0.0156$ ), as well as between teleport and particle trace/dash (IFoV:  $p = 0.004/0.0002$ , OFoV:  $p = 0.0060/0.0012$ ). In the OFoV scenario, we found additionally the portal and walking ( $p = 0.0037$ ) to be rated significantly different.

For the (re-)spotability, we found rather similar significant differences. Specifically, we found them for the IFoV scenario to be between the beam and all other ones except the teleport ( $p = 0.0027/0.0001/0.0051/0.0002/0.0002$ ), slow teleport and particle trace/portal/dash/walking ( $p = 0.0001/0.0003/0.0001/0.0001$ ), between teleport/portal/dash and walking ( $p = 0.0001/0.0044/0.0006$ ), between teleport and portal/dash ( $p = 0.008/0.0002$ ), and between teleport/portal and particle trace ( $p = 0.0002/0.0125$ ). For the OFoV scenario, we also found rather similar significant differences: between beam and slow teleport/walking/dash/particle trace ( $p = 0.0204/0.0013/0.0027/0.0072$ ), slow teleport and walking/dash/particle trace/portal/teleport ( $p = 0.0006/0.0004/0.0005/0.0170/0.0293$ ), particle trace/portal/teleport and walking ( $p = 0.0169/0.0024/0.0027$ ), as well as between teleport and dash/particle trace ( $p = 0.0014/0.0129$ ).

In addition to the questionnaires, we employed controller and eye tracking to

quantify possible differences in the trackability of visualizations. Fig. 4.8 shows representative results. Regarding the controller tracking, the particle trace (i.e., IFoV:  $M = 79.4, Mdn. = 85.8, SD = 19.4$ ), dash (i.e., IFoV:  $M = 83.4, Mdn. = 88.8, SD = 17.3$ ), and walking (i.e., IFoV:  $M = 83.6, Mdn. = 88.3, SD = 17.9$ ) showed the highest results for both scenarios. The data was found to be not normally distributed. The Friedman test and posthoc testing revealed that, in the IFoV scenario, significant differences exist between the slow teleport and dash/walking ( $p = 0.0015/0.0068$ ), between the teleport and particle trace/beam/dash/walking ( $p = 0.021/0.0230.0003/0.0005$ ), between portal and particle trace ( $p = 0.0464$ ), also between beam/portal and walking ( $p = 0.03/0.0034$ ), and between portal and dash ( $p = 0.0031$ ). We got rather similar results for the OFoV scenario: Significant differences were found between the slow teleport and dash/walking ( $p = 0.0111/0.0329$ ), teleport and particle trace/dash/walking ( $p = 0.0046/0.0002/0.0034$ ), also between beam and dash ( $p = 0.0091$ ), and portal and dash/walking ( $p = 0.0046/0.0042$ ). For the gaze tracking, the hit ratio was generally higher, and the differences between visualizations were smaller. However, walking (i.e., IFoV:  $M = 88.4, Mdn. = 93.76, SD = 11.81$ ), dash (i.e., IFoV:  $M = 86.6, Mdn. = 93.78, SD = 14.8$ ) and particle trace (i.e., IFoV:  $M = 85.1, Mdn. = 90.59, SD = 15.15$ ) still ranked the highest. Significant differences were found between: slow teleport and dash/walking (IFoV:  $p = 0.0028/0.0002$ , OFoV:  $p = 0.0211/0.0211$ ), teleport and dash/walking (IFoV:  $p = 0.0193/0.0056$ , OFoV:  $p = 0.0008/0.0002$ ), and between portal and walking (IFoV:  $p = 0.0176$ , OFoV:  $p = 0.0007$ ). In the OFoV scenario, we got additionally: teleport/beam and particle trace ( $p = 0.0034/0.0101$ ), beam and dash/walking ( $p = 0.0031/0.0004$ ), and portal and dash ( $p = 0.0042$ ).

**Perceived Speed and User Preference** The standard teleport was perceived as the quickest for both scenarios (i.e., IFoV:  $M = 6, Mdn. = 6, SD = 1.39$ ), while the slow teleport (i.e., IFoV:  $M = 1.65, Mdn. = 1, SD = 2.04$ ) and particle trace (i.e., IFoV:  $M = 1.85, Mdn. = 2, SD = 1.49$ ) were perceived as the slowest, see Fig. 4.9. The data was not normally distributed. Significant differences were found between (IFoV) slow teleport/particle trace/beam/portal/dash and teleport ( $p = 0.0002/0.0012/0.0006/0.0168/0.017$ ), and between particle trace and walking ( $p = 0.0239$ ). For the OFoV scenario, we found them to be between slow teleport/particle trace/teleport and beam ( $p = 0.0196/0.0252/0.0003$ ), slow teleport/walking/dash/particle trace/portal and teleport ( $p = 0.0002/0.0004/0.0004/0.0002/0.0004$ ).

We found the walking visualization to be the most preferred one for both scenarios (i.e., IFoV:  $M = 4.38, Mdn. = 5, SD = 1.88$ ), and the slow teleport (i.e., IFoV:  $M = 1.31, Mdn. = 1, SD = 1.32$ ) and dash (i.e., IFoV:  $M = 1.42, Mdn. = 1, SD = 1.68$ ) the least preferred ones, see Fig. 4.10. The data was not normally distributed. We found significant differences between (IFoV) the slow teleport/teleport/dash and beam ( $p = 0.0011/0.0425/0.0058$ ), between slow teleport and particle/portal/walking ( $p = 0.009/0.008/0.0045$ ), and between the dash and particle/walking ( $p = 0.0424/0.0004$ ). For the OFoV scenario, we found them to be between slow teleport and beam/walking/particle trace/portal ( $p = 0.0012/0.0003/0.0030/0.0142$ ), between dash/teleport and walking ( $p = 0.0002/0.0114$ ), and between dash and particle trace ( $p = 0.0032$ ).

We did also compare the individual results between the scenarios and experiment variants. For the former, we found the differences to be generally very small and not statistically significant, see for a representative example Figure 4.11 (top). Between the experiment variants, the differences were more noticeable, although they were still mostly not statistically significant, see for example Fig. 4.11 (bottom).

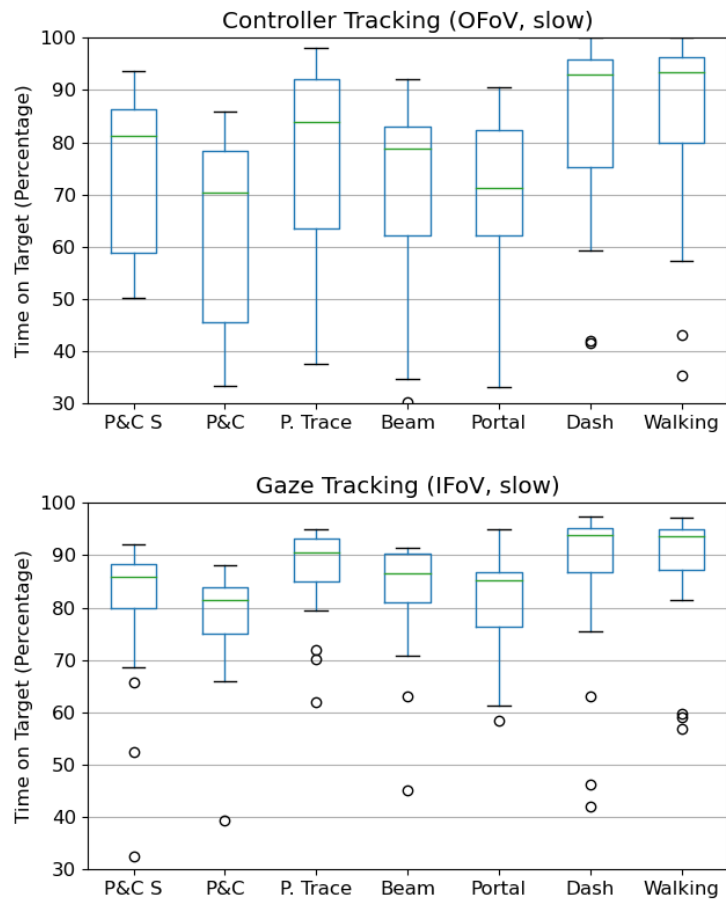


FIGURE 4.8: Controller (top) and eye tracking (bottom) hit rate results in the out-of-FoV/in-FoV scenario and slow variant. The continuous walking, dash, and particle trace visualizations get the highest rates. However, for gaze tracking the hit rates are generally higher and differences smaller.

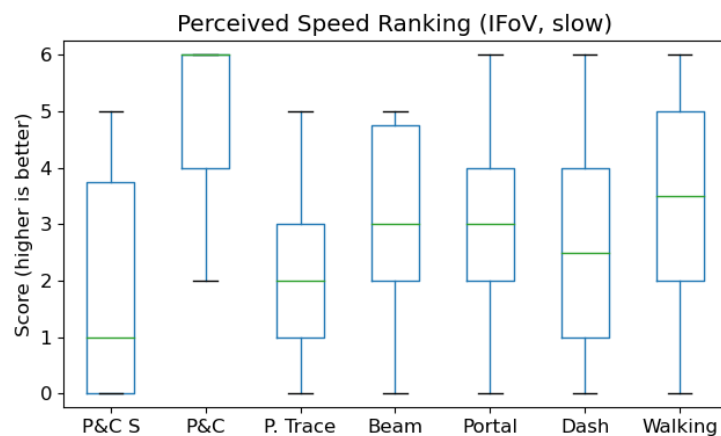


FIGURE 4.9: Ranking results for perceived speed in the in-FoV scenario and slow variant. The standard teleport is rated highest while the slow teleport is rated lowest.



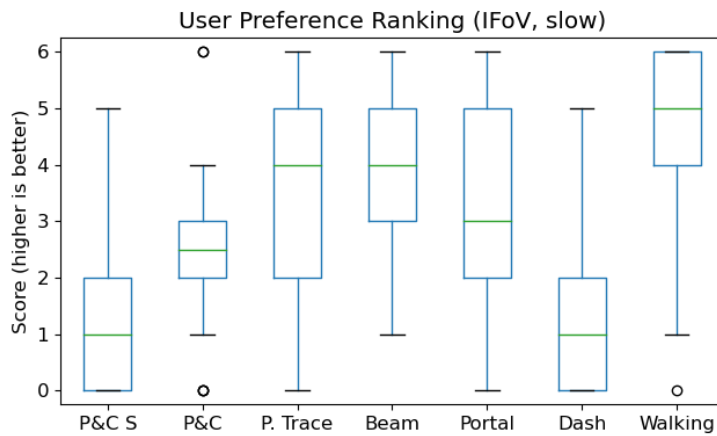


FIGURE 4.10: User preference ranking results in the in-FoV scenario and slow variant. Walking is rated highest, and both teleports and the dash are the lowest.

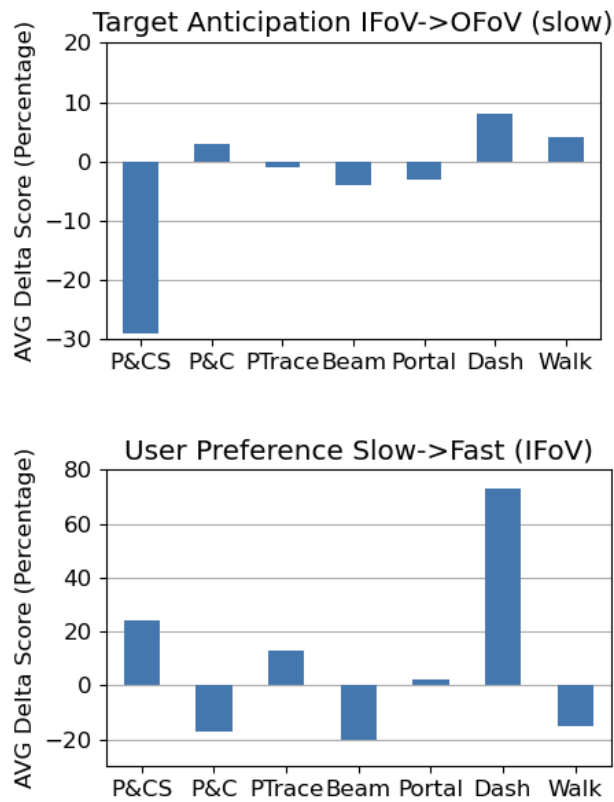


FIGURE 4.11: Relative delta scores of the visualizations between the IFoV/OFoV scenario (top) and slow/fast variant (bottom) In the former, the differences are mostly small but continuous visualizations tend to perform better. In the latter, one can observe more noticeable differences leading to an equalizing effect (e.g., walk decreasing, dash increasing).

TABLE 4.4: All data (mean, median, standard deviation) of the IFoV(I)/OFoV(O) scenarios and slow(s)/fast(f) experiment variants.

Property	P&C S			P&C			P. Trace			Beam			Portal			Dash			Walking		
	M	Mdn.	SD	M	Mdn.	SD	M	Mdn.	SD	M	Mdn.	SD	M	Mdn.	SD	M	Mdn.	SD	M	Mdn.	SD
Social Pres. I/s	3.25	3.50	1.45	3.44	3.75	1.54	3.63	4.00	1.57	3.58	3.75	1.54	3.56	3.75	1.36	3.25	3.50	1.48	4.42	4.75	1.87
Social Pres. O/s	3.37	3.25	1.41	3.62	3.50	1.42	3.85	3.75	1.59	3.69	3.50	1.46	3.80	4.00	1.46	2.98	2.50	1.50	4.31	4.50	1.73
Social Pres. I/f	3.29	3.00	1.63	3.50	3.25	1.50	3.77	3.50	1.63	3.62	4.00	1.56	3.73	3.50	1.74	3.23	2.75	1.63	4.08	4.00	1.71
Social Pres. O/f	3.50	3.00	1.78	3.33	3.00	1.65	3.67	3.50	1.83	3.58	3.50	1.67	3.46	3.50	1.74	3.63	3.50	1.80	4.17	4.25	1.65
Spatial Pres. I/s	4.38	4.50	1.49	4.46	4.50	1.44	4.44	4.50	1.58	4.60	4.50	1.36	4.35	4.00	1.43	4.54	4.50	1.22	5.19	5.50	1.32
Spatial Pres. O/s	4.50	4.50	1.41	4.62	4.75	1.40	4.73	4.75	1.29	4.48	4.50	1.34	4.46	4.50	1.29	4.23	4.25	1.57	5.21	5.25	1.30
Spatial Pres. I/f	4.23	4.25	1.50	4.38	4.50	1.40	4.33	4.75	1.60	4.31	4.50	1.52	4.08	4.00	1.60	4.02	4.00	1.68	4.54	5.00	1.31
Spatial Pres. O/f	4.42	4.50	1.53	4.12	4.50	1.65	4.25	4.50	1.77	4.29	4.50	1.43	4.08	4.25	1.57	4.21	4.50	1.54	4.75	5.00	1.47
Plausibility I/s	1.77	2.00	1.42	2.54	2.50	1.77	2.81	3.00	1.77	3.42	3.00	1.33	2.92	3.00	2.04	2.12	1.50	2.08	5.42	6.00	1.27
Plausibility O/s	1.81	1.00	1.52	2.12	2.00	1.66	3.15	3.00	1.76	3.23	4.00	1.82	3.00	3.00	1.77	2.23	2.00	1.90	5.46	6.00	1.27
Plausibility I/f	1.54	2.00	1.33	2.42	2.00	1.86	3.27	4.00	1.95	2.81	3.00	1.63	3.12	3.00	1.93	3.08	3.50	1.98	4.77	6.00	1.97
Plausibility O/f	2.15	2.00	1.52	2.35	2.00	1.94	3.12	3.50	1.99	3.23	4.00	1.58	2.54	3.00	2.04	2.88	2.00	1.97	4.73	6.00	1.99
Intuitiveness I/s	1.42	1.50	1.21	1.81	1.00	1.58	3.81	4.00	1.50	3.46	4.00	1.39	2.96	3.00	1.99	2.19	2.00	1.96	5.35	6.00	1.35
Intuitiveness O/s	1.46	1.00	1.39	2.08	2.00	1.81	3.58	4.00	1.72	2.77	3.00	1.63	3.23	3.00	1.70	2.35	2.00	1.87	5.54	6.00	0.95
Intuitiveness I/f	1.62	2.00	1.47	2.27	2.00	1.87	3.38	4.00	1.98	3.08	3.50	1.62	3.27	3.50	2.03	2.62	2.50	1.77	4.77	6.00	1.86
Intuitiveness O/f	2.19	2.00	1.50	2.12	2.00	1.68	3.08	4.00	2.02	2.77	3.00	1.63	2.35	2.00	1.96	3.27	4.00	1.89	5.23	6.00	1.63
Target Anticip. I/s	1.08	1.00	1.20	1.50	1.00	1.61	4.38	4.50	1.39	2.00	2.00	1.06	3.19	3.00	2.08	3.69	4.00	1.38	5.15	6.00	1.12
Target Anticip. O/s	0.77	0.00	1.07	1.54	1.00	1.50	4.35	4.00	1.47	1.92	2.00	1.02	3.08	3.00	1.70	4.00	4.00	1.20	5.35	6.00	0.98
Target Anticip. I/f	0.88	1.00	1.31	1.31	1.00	1.52	4.19	4.00	1.55	2.35	2.00	1.32	3.35	3.00	1.60	3.77	4.00	1.48	5.15	6.00	1.05
Target Anticip. O/f	0.92	1.00	0.98	1.38	1.00	1.53	4.38	4.00	1.42	2.38	2.00	1.27	2.81	3.00	1.72	4.04	5.00	1.40	5.08	6.00	1.44
Ease to Spot I/s	0.46	0.00	0.65	1.15	1.00	1.12	4.46	4.00	0.95	1.85	2.00	0.88	3.23	3.00	1.34	4.38	5.00	0.80	5.46	6.00	1.07
Ease to Spot O/s	0.77	0.00	1.27	1.65	1.00	1.47	3.96	4.00	1.22	2.15	2.00	1.22	2.69	3.00	1.62	4.46	5.00	1.14	5.31	6.00	1.35
Ease to Spot I/f	0.96	1.00	0.92	1.31	1.00	1.46	4.31	4.50	1.64	2.00	2.00	1.33	3.38	3.00	1.58	3.88	4.00	1.53	5.15	5.00	0.97
Ease to Spot O/f	1.23	1.00	1.48	1.27	1.00	1.34	4.15	4.00	1.38	2.15	2.00	1.26	2.42	2.00	1.65	4.19	4.00	1.10	5.58	6.00	0.70
Perceived Speed I/s	1.65	1.00	2.04	5.12	6.00	1.40	1.85	2.00	1.49	3.00	3.00	1.70	3.19	3.00	1.67	2.73	2.50	1.76	3.46	3.50	1.94
Perceived Speed O/s	1.46	0.00	2.16	5.69	6.00	0.55	1.65	1.50	1.38	3.46	4.00	1.50	2.96	3.00	1.40	2.77	3.00	1.68	3.00	3.00	1.74
Perceived Speed I/f	2.69	2.00	1.95	5.19	6.00	1.70	1.85	1.50	1.71	3.65	4.00	1.47	3.15	3.00	1.26	2.31	2.00	1.81	2.15	1.50	2.07
Perceived Speed O/f	2.19	2.00	1.94	4.85	6.00	1.95	2.19	2.00	1.67	3.35	3.50	1.47	3.08	3.00	1.92	2.81	3.00	1.98	2.54	3.00	1.92
User Preference I/s	1.31	1.00	1.32	2.58	2.50	1.55	3.65	4.00	1.90	4.12	4.00	1.34	3.54	3.00	1.86	1.42	1.00	1.68	4.38	5.00	1.88
User Preference O/s	1.23	1.00	1.31	2.54	2.00	1.61	4.08	4.50	1.76	3.50	4.00	1.48	3.38	3.00	1.92	1.46	1.00	1.63	4.81	5.00	1.50
User Preference I/f	1.62	1.00	1.44	2.15	2.00	1.85	4.12	5.00	2.01	3.31	3.50	1.85	3.62	4.00	2.02	2.46	2.00	1.70	3.73	4.00	1.95
User Preference O/f	1.69	1.00	1.49	2.96	3.00	1.87	3.58	4.00	2.19	3.46	3.00	1.82	3.27	3.00	2.03	2.19	2.00	1.70	3.85	4.00	2.11
Cont. Tracking I/s	73.40	79.20	19.00	70.50	75.00	15.40	79.40	85.80	19.40	77.00	81.60	15.00	72.20	75.80	13.00	83.40	88.80	17.30	83.60	88.30	17.90
Cont. Tracking O/s	72.70	81.40	19.80	63.90	70.50	18.00	77.40	84.00	18.70	69.50	78.80	18.90	69.30	71.30	16.90	83.90	93.00	16.70	82.70	93.50	23.80
Cont. Tracking I/f	79.10	80.90	10.90	71.00	71.70	10.00	77.80	82.30	16.70	71.80	73.70	14.30	70.20	74.60	13.10	76.40	77.70	11.80	78.80	79.40	8.60
Cont. Tracking O/f	69.60	77.60	24.70	63.80	70.90	21.30	64.00	72.20	24.30	66.00	69.30	19.50	64.40	67.50	16.80	73.70	80.70	18.60	72.10	78.50	20.40
Gaze Tracking I/s	79.00	86.00	17.30	78.50	81.40	9.70	85.10	90.60	15.20	83.40	86.70	10.50	80.80	85.20	9.00	86.60	93.80	14.80	88.40	93.80	11.80
Gaze Tracking O/s	81.60	88.90	17.70	72.70	79.30	15.60	87.90	95.70	16.90	80.00	84.80	16.60	80.70	84.50	13.70	94.10	97.50	10.30	93.10	96.80	10.90
Gaze Tracking I/f	87.40	89.00	6.20	81.10	83.80	5.30	86.10	88.60	6.90	84.80	85.90	5.90	81.50	83.50	8.90	87.00	87.90	4.10	86.70	87.90	4.60
Gaze Tracking O/f	83.70	88.60	14.60	78.20	82.60	12.90	83.20	87.20	14.90	80.20	84.80	14.00	77.20	79.70	9.90	88.30	88.90	7.10	87.10	88.60	7.20

In the following, we depict tables with all the data that we gathered. Specifically, we list all medians, means, and standard deviations for all visualizations and scenarios/variants. We also list all visualization pairs for which we found significant differences.

## 4.6 Discussion

In this section, we interpret the findings of our study, especially with regard to our hypothesis.

The results about cybersickness are very good, we did not have or see any issues in this regard.

Our results show, that the walking visualization is consistently rated higher in social presence than the others, including the teleport, and the difference to the latter to be statistically significant. For the spatial presence, the results were closer together, but, still, the walking visualization was always rated highest. In the more critical OFoV scenario, significant differences to many other visualizations were found, while in the IFoV scenario, this threshold was mostly not reached, although the tendencies were present, too. These results indicate that the visualization of the teleport does have a positive effect on spatial as well as social presence. Thus, we can partly confirm our hypotheses  $H_3$  and  $H_4$ . We can only confirm it partly, as this seems to be highly dependent on the actual visualization, and just having one not necessarily

TABLE 4.5: Plausibility: All statistical different pairs of the IFoV/OFoV scenarios and slow/fast experiment variants.

IFoV/Slow			OFoV/Slow			IFoV/Fast			OFoV/Fast		
Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p
Beam	Walking	0.0060	Beam	Walking	0.0143	P&C S	Walking	0.001	P&C S	Walking	0.0069
P&C S	Walking	0.0002	P&C S	Walking	0.0002	Walking	Dash	0.0045	Walking	Dash	0.0252
Walking	Dash	0.0001	Walking	Dash	0.0002				Walking	P&C	0.0393
Walking	P. Trace	0.0061	Walking	P. Trace	0.0191						
Walking	Portal	0.0169	Walking	Portal	0.0119						
Walking	P&C	0.0010	Walking	P&C	0.0003						

TABLE 4.6: Intuitiveness: All statistical different pairs of the IFoV/OFoV scenarios and slow/fast experiment variants.

IFoV/Slow			OFoV/Slow			IFoV/Fast			OFoV/Fast		
Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p
Beam	P&C S	0.0028	Beam	Walking	0.0004	P&C S	Walking	0.0024	Beam	Walking	0.0086
Beam	Walking	0.0184	P&C S	Walking	0.0002	Walking	Dash	0.0007	P&C S	Walking	0.0007
Beam	P&C	0.0374	P&C S	P. Trace	0.0436	Walking	P&C	0.0252	Walking	Dash	0.0016
P&C S	Walking	0.0002	Walking	Dash	0.0004				Walking	Portal	0.0159
P&C S	P. Trace	0.0028	Walking	P. Trace	0.0091				Walking	P&C	0.0029
Walking	Dash	0.0003	Walking	Portal	0.0065						
Walking	Portal	0.0201	Walking	P&C	0.0004						
Walking	P&C	0.0003									

TABLE 4.7: Target Anticipation: All statistical different pairs of the IFoV/OFoV scenarios and slow/fast experiment variants.

IFoV/Slow			OFoV/Slow			IFoV/Fast			OFoV/Fast		
Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p
Beam	Walking	0.0003	Beam	Walking	0.0001	Beam	P&C S	0.0106	Beam	P&C S	0.0071
Beam	Dash	0.0138	Beam	Dash	0.0046	Beam	Walking	0.0005	Beam	Walking	0.0014
Beam	P. Trace	0.0014	Beam	P. Trace	0.0007	Beam	P. Trace	0.0144	Beam	Dash	0.0376
P&C S	Walking	0.0002	P&C S	Walking	0.0002	P&C S	Walking	0.0002	Beam	P. Trace	0.0020
P&C S	Dash	0.0017	P&C S	Dash	0.0002	P&C S	Dash	0.0020	P&C S	Walking	0.0002
P&C S	P. Trace	0.0003	P&C S	P. Trace	0.0008	P&C S	P. Trace	0.0024	P&C S	Dash	0.0003
P&C S	Portal	0.0387	P&C S	Portal	0.0034	P&C S	Portal	0.0023	P&C S	P. Trace	0.0009
Walking	Dash	0.0007	Walking	Dash	0.0156	Walking	Dash	0.0026	P&C S	Portal	0.0164
Walking	P&C	0.0002	Walking	Portal	0.0037	Walking	Portal	0.0237	Walking	Dash	0.0477
Dash	P&C	0.0195	Walking	P&C	0.0003	Walking	P&C	0.0003	Walking	Portal	0.0160
P. Trace	P&C	0.0040	Dash	P&C	0.0012	Dash	P&C	0.0041	Walking	P&C	0.0008
			P. Trace	P&C	0.0060	P. Trace	P&C	0.0036	Dash	P&C	0.0036
						Portal	P&C	0.0075	P. Trace	Portal	0.0493
									P. Trace	P&C	0.0012



TABLE 4.11: Social presence: All statistical different pairs of the IFoV/OFoV scenarios and slow/fast experiment variants.

IFoV/Slow			OFoV/Slow			IFoV/Fast			OFoV/Fast		
Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p
P&C	S Walking	0.0038	P&C	S Walking	0.0024	P&C	S P. Trace	0.0305	P&C	Walking	0.0152
P&C	Walking	0.0445	P. Trace	Dash	0.0437	P&C	S Walking	0.0101			
Dash	Walking	0.0181	Beam	Dash	0.0326	P. Trace	Dash	0.0201			
			Portal	Dash	0.0477	Dash	Walking	0.0032			
			Dash	Walking	0.0030						

TABLE 4.12: Spatial presence: All statistical different pairs of the IFoV/OFoV scenarios and slow/fast experiment variants.

IFoV/Slow			OFoV/Slow			IFoV/Fast			OFoV/Fast		
Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p
Dash	Walking	0.0105	P&C	S Walking	0.0273				Portal	Walking	0.0058
			P. Trace	Walking	0.0202						
			Beam	Walking	0.0358						
			Portal	Walking	0.0184						
			Dash	Walking	0.0100						

TABLE 4.13: Controller tracking: All statistical different pairs of the IFoV/OFoV scenarios and slow/fast experiment variants.

IFoV/Slow			OFoV/Slow			IFoV/Fast			OFoV/Fast		
Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p
P&C	S Dash	0.0015	P&C	S Dash	0.0111	P&C	S P&C	0.0231	Beam	Dash	0.0464
P&C	S Walking	0.0068	P&C	S Walking	0.0329	P&C	S Portal	0.0359	Portal	Walking	0.0359
P&C	P. Trace	0.0211	P&C	P. Trace	0.0046	P&C	P. Trace	0.0083			
P&C	Beam	0.0231	P&C	Dash	0.0002	P&C	Walking	0.0068			
P&C	Dash	0.0003	P&C	Walking	0.0034	P. Trace	Portal	0.0426			
P&C	Walking	0.0005	Beam	Dash	0.0091	Portal	P. Trace	0.0038			
P. Trace	Portal	0.0464	Beam	Walking	0.0046	Portal	P. Trace	0.0038			
Beam	Walking	0.0301	Portal	Dash	0.0042	Portal	P. Trace	0.0038			
Portal	Dash	0.0031				Portal	P. Trace	0.0038			
Portal	Walking	0.0034				Portal	P. Trace	0.0038			

TABLE 4.14: Gaze Tracking: All statistical different pairs of the IFoV/OFoV scenarios and slow/fast experiment variants.

IFoV/Slow			OFoV/Slow			IFoV/Fast			OFoV/Fast		
Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p	Vis. 1	Vis. 2	p
P&C	S Dash	0.0028	P&C	S Dash	0.0211	P&C	S P&C	0.0028	P&C	Dash	0.0028
P&C	S Walking	0.0002	P&C	S Walking	0.0211	P&C	S Portal	0.0464	P&C	Walking	0.0146
P&C	Dash	0.0193	P&C	P. Trace	0.0034	P&C	P. Trace	0.0091	Beam	Dash	0.0464
P&C	Walking	0.0056	P&C	Dash	0.0008	P&C	Dash	0.0062	Portal	Dash	0.0004
Portal	Walking	0.0176	P&C	Walking	0.0002	P&C	Walking	0.0003	Portal	Walking	0.0034
			P. Trace	Beam	0.0101	P. Trace	Portal	0.0329			
			Beam	Dash	0.0031						
			Beam	Walking	0.0004						
			Portal	Dash	0.0042						
			Portal	Walking	0.0007						

improves the presence. Generally, the spatial presence was consistently higher than the social one, which is understandable, as it might be more affected by the teleportation of the other user. Another reason could be that the social presence was lower to begin with, as there is no actual social interaction in the experiment. A more complex, dynamic, and engaging environment and setup may be needed to induce more social as well as spatial presence, to begin with. Then, possibly, we could see also more differences between the other visualizations. User feedback we collected after the experiment, points in the same direction, as 2 participants stated that a more detailed environment would be helpful for presence and 8 participants did remark that the avatar's look, animation, or interaction with each other was lacking (e.g.: "dead eyes", "animations unnatural", "no eye contact", "no handshake"). The comments about the avatar and its animation are interesting, as we recorded them with motion tracking. Possibly, the actual physical transition between standing and walking was too abrupt and unnatural, as it was specifically intended for the teleport action. Another reason is probably that even the slow experiment variant was sped up from the actual recording.

Regarding the plausibility and intuitiveness of the locomotion process, our results clearly show that the walking visualization is consistently rated the highest, most often significantly, compared to all other ones. Moreover, the standard and delayed teleport, as well as the dash were consistently rated the lowest. These results confirm our expectations that a full walking animation is perceived as very plausible and intuitive while vanishing and emerging without any motion cues (teleport) is absolutely not. That the dash is rated low in this regard is understandable, too, as just translating a static avatar can look weird or jarring. With these results, we can confirm our hypotheses  $H_1$  and  $H_2$  that teleport visualizations, in general, make the locomotion process more intuitively understandable and plausible, thus, reduce potentially occurring confusion. The degree to which the visualizations improve plausibility and intuitiveness is, again, clearly dependant on the visualization itself.

The results about target anticipation and the ease of re-spotting the avatar after the locomotion show, as expected, that continuous visualizations perform significantly better than discontinuous ones, especially compared to the visualization-less teleports. The walking animation performed best overall, and, interestingly, the portal was the best out of the discontinuous ones. The latter result may be because the portal itself gives an indication of the direction of locomotion, while the beam and the teleport variants do not give any hints. With these results, we can confirm our hypotheses  $H_5$  and  $H_6$ , that continuous teleport visualizations are advantageous for tracking and relocating a teleporting user, thus, increasing spatial awareness. These findings are in line with the ones by Freiwald et al. [113]. These results moreover implicitly reinforce hypothesis  $H_1$  and  $H_2$ , as easier tracking and increased spatial awareness of the other user's location should reduce confusion, too.

The quantitative results of the controller and gaze tracking, generally, paint the same picture and confirm the results regarding target anticipation and (re-)spotting. The continuous visualizations had the highest hit rates, often significantly higher, reinforcing hypotheses  $H_5$  and  $H_6$ . Generally, the hit rates were higher for the gaze tracking than the ones from controller tracking and showed smaller differences between visualizations. This is understandable, as it is plausible that it is faster to change the eye gaze direction to the target location than the handheld laser pointer. Arguably, it is also easier to follow the movements of the avatar with the gaze. Interestingly, with this data, the portal performed worse than the beam, while the qualitative results were the opposite. This may be due to the portal object distracting from the avatar.

The standard teleport was perceived as the quickest and most often rated significantly higher regarding perceived speed than the other visualizations. This is expected, as it is the only one which is actually performed instantly. The delayed teleport, however, which takes the same time as the others, was consistently rated as the slowest. This may be because of the absolute lack of any visuals which may make the time until reappearance appear longer. The particle trace performed poorly, too, which may be due to the nature of the effect itself and the slow-rising movement of the individual particles. In the OFoV scenario, for instance, both these methods were rated significantly lower than the beam.

Our results show also that the walking animation is clearly the most preferred visualization, and the teleport variants and the dash were rated lowest. Moreover, the differences in ratings were often significantly high. This result, together with the fact that the walking animation was rated best in all other tested categories, except perceived speed, too, more than confirms our hypothesis  $H_7$ , that this is the best teleport visualization. Our results confirm the findings by Freiwald et al. [113], that continuous visualizations (walking/particle trail (us), walking/dissolve (them)) are significantly more preferred than the standard teleport.

#### 4.6.1 Comparison of IFoV and OFoV Scenarios

Interestingly, the results for the easier “in the field of view” and more critical “out of the field of view” scenarios are more similar than we expected. For instance, the tracked hit rates decreased in the OFoV scenario, especially for discontinuous methods such as the teleport versions. However, the differences were not as high as we would have expected. Also, the continuous visualizations tended to perform better, relative to the others, regarding the target anticipation in the OFoV scenario, which is reasonable. See for example Figure 4.9 (top). However, again, the difference between scenarios is smaller than assumed. This may indicate that the standard, visualization-less teleport is problematic in even simpler scenarios such as our IFoV scenario, which makes the visualizations even more important.

#### 4.6.2 Comparison of Slow and Fast Experiment

As to the comparison of the slow and fast experiment variants, we did observe that the presence scores were mostly slightly lower in the faster experiment variant. We found the tracked hit rates to be generally lower, too, in the fast variant. The hit rates of the continuous visualizations, which were higher previously, were the most affected while the hit rates of the teleport variants were less affected. Thus, in the faster variant, the hit rates between the visualizations were closer together. Interestingly, the ratings about target anticipation and (re-)spotability stayed roughly the same. Regarding the plausibility and intuitiveness, however, we found that the advantages of the walking visualization decreased while the dash got better, relatively (the teleport variants to a lesser degree, too). Thus, we observe a homogenizing effect again. The continuous visualizations’ scores regarding perceived speed decreased (mostly the walking animation), while the discontinuous ones mostly increased, at least in the IFoV scenario. In contrast to the slow experiment variant, we found the particle trace to have the highest scores regarding user preference in the IFoV scenario, as it was rated higher and the walking animation lower, see Figure 4.9 (bottom). The rankings for the portal and especially the dash increased, too. We see the same tendencies for the dash and walking visualizations in the OFoV scenario, making the particle trace and walking the most preferred visualizations overall. To

summarize, the advantages of the visualizations decreased in the fast experiment variant, and the differences between the visualizations shrunk, e.g. dash catching up and walking coming down. We can therefore state that the distance scalability of the visualizations varies. However, we cannot confirm our hypothesis  $H_8$  that continuous ones are principally worse.

The fact that the walking animation loses so much regarding user preference, plausibility, and intuitiveness while the dash's ratings increases, is understandable, as the walking animation is more sped up, thus looking less natural. We find it also plausible, that the continuous visualizations fare worse in tracked hit rates, as these had the highest ratings in the slow variant, and it gets arguably harder for the observer to track the avatar/visualization when moving faster. Moreover, we see that the higher the locomotion speed, the lesser the advantages of the visualizations, especially when they get more unnatural (i.e., the simply sped-up walk animation). We find this logical, as there is less time that the teleporting user is not visible to the observer, and there is less time for a continuous visualization to show the motion cues.

We refer, as mentioned earlier, to Appendix B for all plots, as it would have been too overwhelming to include them all directly in the result section.

## 4.7 Limitations

We opted for a high fidelity but also a minimalistic virtual environment, decided to have a single pre-recorded character teleporting through the scene, and limit ourselves to a single observer with the simple task of just observing and tracking the other person moving. We did this, in order to limit distractions of the participants, focus on the primary question of how the visualizations affect the observer, and minimize confounding effects between the users, the environment, and the scenario. This setup, however, may have not been engaging and long enough to build high levels of presence, reducing the possible positive effects of the teleport visualizations. With our design choices for this first study, we are, naturally, also unable to fully replicate and investigate actual multi-user conditions with multiple users teleporting and multiple users observing. Having multiple users teleporting at the same time could possibly alter the requirements and suitability of the individual visualizations, as paths could be intersecting. Similarly, in a more complex and dynamic environment, other dynamically moving objects could get in the way, or target destinations may be not conventionally reachable. This would make real-time path planning and visualization and possibly more general visualizations necessary. Also, having multiple elaborate but equally looking teleport visualizations from various users at the same time could make the scene more distracting and unclear again. Moreover, more complex, collaborative tasks than just one-way observation would be highly interesting to investigate, too, as then it may be more relevant if the observer recognizes the teleport as such and if and when he is aware of the destination and traveling path.

Another limitation of our current work is that we, for now, focused solely on the visualizations' effects on the observers but not on the teleporting user himself. For instance, the teleporting user probably will prefer a quick teleportation process to minimize waiting time, while observers, in contrast, would prefer to have some duration to observe motion cues and increase immersion.



## 4.8 Conclusions and Future Work

With this work, we presented a user study to investigate suitable visualizations to depict the deliberate act of teleportation to observers in multi-user VR. The goal was to evaluate if they enhance spatial awareness, reduce confusion and, thus, help to retain as much presence as possible. For our study, we implemented seven different visualization techniques, continuous and non-continuous ones, into a virtual environment using the Unreal Engine 4. In our experiment, we compared the visualizations and their effects on observers. The properties we examined were perceived social presence, spatial presence, confusion, distance scalability, and spatial awareness. We found that teleport visualizations can have significant positive effects on social as well as spatial presence, but do not necessarily have to. Continuous visualizations significantly increased spatial awareness. Moreover, various visualizations were rated significantly higher regarding plausibility and intuitiveness, which indicates less confusion. The results show that the type of visualization affects the distance scalability and that a walking animation is the overall best-performing, user-preferred visualization. These findings not only hold when teleporting out of the observer's view but also when the start, path, and target are all in view. On the other hand, the advantages of the visualizations decrease, when increasing the teleportation/locomotion speed.

In the future, it would be interesting to conduct a similar study in a more complex, interactive environment with multiple users teleporting and observing and with collaborative tasks between users. With this, users would have a more engaging experience that induces a higher presence and one would be able to provide a deeper investigation of the advanced requirements and effects of the visualizations and the issues arising from teleportation in multi-user VR environments. Furthermore, we would find it interesting to investigate the depth perception of the visualizations and how the teleportation visualizations affect the teleporting user's presence and usability, as well as how this changes with different distances that have to be covered.



## Chapter 5

# Large-scale Procedural Terrain Generation for VR Environments

Previously, we considered how to reconstruct and render presence-inducing avatars in multi-user VR. However, (multi-user) VR applications usually also need convincing, high-quality 3D environments to convey a feeling of presence. In some use cases, RGB-D sensor-based live reconstructions are employed to visualize 3D environments or parts thereof. As mentioned in Section 1.1, most commonly, the 3D environments consist of manually modeled meshes. However, in some areas, manually modeling the environment is not feasible, as large, detailed landscapes are required. One example of such a scenario is (collaborative) VR testbeds that simulate the navigation and interaction of unmanned vehicles and robots with different terrains. Another example is environmental simulation environments. In these example scenarios, accurate, high-quality environments and terrains not only serve to provide a sense of presence but are also required for accurate simulations. To generate those terrains efficiently, procedural terrain generation algorithms come into play.

To also cover the aspect of procedurally generating these 3D environments that can be used in such VR applications, we present in this section two advanced PTG systems and the idea for a third one (including theoretical considerations and relevant related work) that serves to broaden the horizon. First, in Section 5.1, we propose a broadly applicable pipeline for the procedural generation of large-scale multi-biome landscapes. Then, in Section 5.2, a system is presented that tackles the more advanced challenge of procedurally generating landscapes with naturally distributed water bodies such as rivers. Lastly, Section 5.3 presents the previously mentioned concepts, considerations, and relevant related work for a system that would be able to procedurally create extraterrestrial planetary surfaces, specifically, lookalikes of example DEM input images that feature the same characteristics. The intended use case would be the application of VR testbeds for the aforementioned simulation of unmanned vehicles.

## 5.1 AutoBiomes: Procedural Generation of Multi-Biome Landscapes

In this section, we want to present a broadly applicable terrain generation system that can be used to create large-scale, detailed, diverse virtual environments for a wide array of VR applications such as virtual testbeds. Although (semi-)automatic procedural terrain generation is a popular and widely used technique, most of the

existing methods are usually highly specialized for certain terrain types, and especially the procedural generation of landscapes composed of different biomes is a scarcely explored topic.

We propose a novel system, called AutoBiomes, which is capable of efficiently creating vast terrains with plausible biome distributions and therefore different spatial characteristics. The main idea is to combine several synthetic procedural terrain generation techniques with digital elevation models and a simplified climate simulation. Moreover, we include an easy-to-use asset placement component that creates complex multi-object distributions. Our system relies on a pipeline approach with a major focus on usability. Our results show that our system allows the fast creation of realistic-looking terrains that can easily be integrated into VR applications.

The work presented in this section is based on our published paper PC4 in Appendix A.

### 5.1.1 Introduction

The ever-rising demand for bigger and more complex virtual 3D worlds poses a challenge for designers to create and fill them with life. There is a broad range of applications for huge and realistic 3D landscapes, e.g., computer games, movies, and simulations. With the rising accessibility of HMDs, there is also an increasing opportunity to explore these worlds in virtual reality in a more immersive environment. We presented some examples in Section 2.3. Generating these worlds manually is a laborious and expensive task [6], therefore extensive research was done in the field of procedural terrain generation. Yet it remains an important topic, as there is still much potential for improvement. Numerous algorithms for PTG have been proposed which can be roughly categorized into three types: synthetic, physics-based, and example-based approaches [144, 127]. Each of these approaches comes with its own strengths and weaknesses. For a brief description, we refer to Section 2.2.6. Generally, most of the currently used methods and terrain generators follow one of the mentioned approaches and emphasize only on a single, very specific use case. Hence, they are hardly capable of satisfying a broader set of requirements [347]. In consequence, it remains a challenge to create a system with a reasonable compromise of the four most essential but mutually contradictory requirements: realism, performance, usability, and flexibility (see Fig. 5.1).

Two other significant factors of creating plausible, detailed 3D worlds received not much attention in the past: the distribution of assets and the generation of landscapes as a combination of different biomes. However, with the rising dimensions of 3D worlds, the interest in landscapes with various characteristics is growing. The procedural distribution of assets faces similar challenges in balancing the requirements as the terrain generation itself and is equally important to create a convincing environment with an organic feel.

Our main contribution is the design and implementation of a PTG system that combines the three main approaches, synthetic, physics-based as well as example-based PTG, and unites the respective advantages to an effective and well-balanced terrain generator. The goal is to generate realistic terrains while keeping simultaneously computation times low and still considering usability and flexibility.

Our focus is not restricted to the generation of huge terrains but covers specifically terrains composed of different biomes, which is a relatively sparsely explored topic with additional challenges. As part of our PTG architecture, we propose an effective biome- and rule-based local-to-global model to populate the terrain with

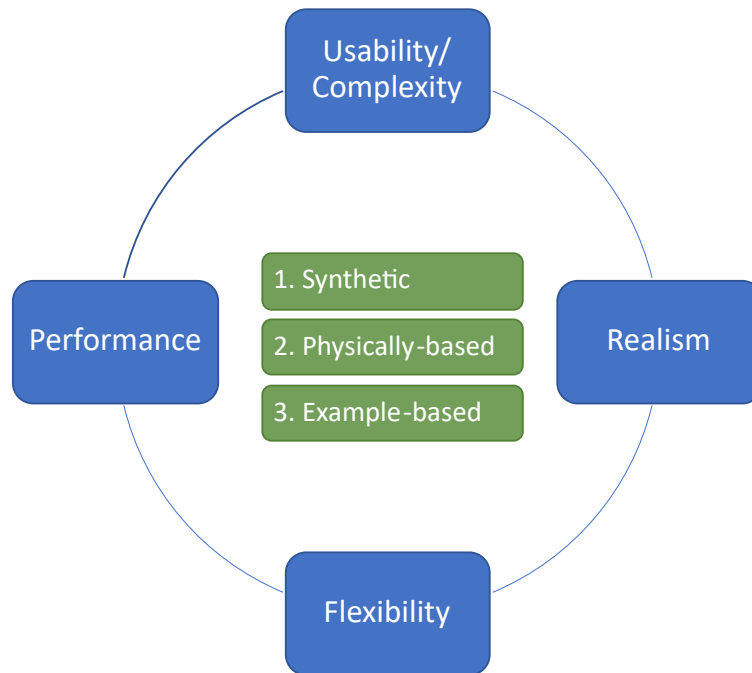


FIGURE 5.1: The three main approaches to PTG and the contradictory requirements of PTG systems.

assets. This component is a vital step to produce a comprehensive solution for creating convincing 3D landscapes.

Finally, our system is implemented in the Unreal Engine and designed to be used completely from within the editor. Optionally, the exported heightmaps can be used in external applications.

### 5.1.2 Related Work

Procedural generation is used since the 1980s and numerous different methods were developed. Noise-based methods belong to the synthetic approach of PTG, one of the oldest and most widely used techniques. Examples of well-known noise functions are Perlin noise by Ken Perlin [278] and his improved version named Simplex noise [279]. More complex results can be achieved by combining multiple instances of noise with different frequencies, called fractal noise. Terrain generation using noise is very popular because it is easy compared to other approaches and the computational effort is low. Drawbacks are the inherently unintuitive way to adjust noise parameters and consequently, the difficulty to create genuinely realistic-looking terrain, as described in [149].

On the other hand, physics-based procedural generation methods have their focus on creating realistic results at the expense of lower computation speed. Very common are erosion algorithms which try to create the terrain by simulating the natural erosion processes. In 1989 Musgrave et al. [254] proposed models for thermal and hydraulic erosion simulation which became the basis for a lot of the subsequent research on this topic. Jákó [157] adapted and improved previous work resulting in more, realistic, versatile and stable results, and presented a faster implementation using the GPU. Another approach is the simulation of fluid dynamics. Most of its techniques either are grid-based, called Eulerian, or particle-based, called

Lagrangian. The leading concept for the latter ones is smoothed particle hydrodynamics, which was well summarized by Ihmsen et al. [151]. Well-known is also the work of Jos Stam, who eventually presented a convincing real-time fluid solver [353] which combined both approaches.

Another concept for PTG techniques is based on using examples, e.g., images or user sketches, and synthesizing terrain according to it. DEMs are digital representations of real ground surfaces, commonly parts of the earth's topography, and can also serve as examples for PTG. Using these DEMs and texture synthesis methods, Zhou et al. [442] presented a system capable of generating realistic-looking terrains if provided with appropriate and detailed data. Not long ago generative neural networks could successfully be applied in the field of PTG. Recently Beckham and Pal [21] and Wulff-Jensen et al. [417] trained deep convolutional generative adversarial networks (DCGANs), developed by Radford et al. [292], on DEMs to create similar looking heightmaps for terrain generation. Similarly, Guérin et al. [135] used conditional generative adversarial networks to create a set of task-specific synthesizers which generate terrain features based on sketches. Gatys et al. [125] also proposed an interesting technique called style transfer where convolutional neural networks learn to combine the artistic style of one image with the main features of arbitrary other images.

A comprehensive overview of all kinds of procedural terrain generation and modeling techniques is given by Galin et al. [119].

In the domain of generating asset distributions, two different concepts can be found. Local-to-global models are based on the individual object instances and by constrained-based placement and simulation of interactions the resulting distribution is determined. Global-to-local models, on the other hand, infer the position of individuals by a beforehand defined distribution. Both Deussen et al. [83] and Lane et al. [199] presented convincing individual-based simulation models to generate plant distributions. A popular distribution to sample objects from is the Poisson distribution, which ensures a minimal distance between samples. Early techniques for Poisson-disk sampling relied on the dart-throwing principle. Jones [167] introduced the combination with a spatial data structure later. Also using spatial subdivision, Gamito and Maddock [123] proposed an accurate and considerably faster algorithm.

### 5.1.3 Proposed Approach

We present a PTG system that combines synthetic, physics- and example-based approaches to produce vast landscapes composed of different biomes and populated with huge amounts of assets. We chose an incremental pipeline design with a focus on high performance to ensure providing the user with quick results. The pipeline currently consists of four individual main steps; each is fully customizable. Direct visualization of each step improves usability and provides a fast, iterative workflow. In case a single step does not meet the user's desires, it can be easily repeated. Additionally, intermediate results are cached to allow the reuse of finished pipeline steps. This system design guarantees the best trade-off between the partly contradictory requirements such as performance, usability, realism, and flexibility.

Figure 5.2 illustrates the individual four steps of our sequential pipeline. The idea is to generate a coarse base terrain using noise functions first, which gets refined with biome-specific details later. To compute realistic biome distributions, we implemented a multiple-step climate simulation, which is carefully simplified to meet the performance requirements while maintaining good results. To add biome-specific terrain details we chose an example-based approach where DEM data is combined

with the previously generated base terrain. Finally, it is possible to generate asset distributions following a rule-based local-to-global model.

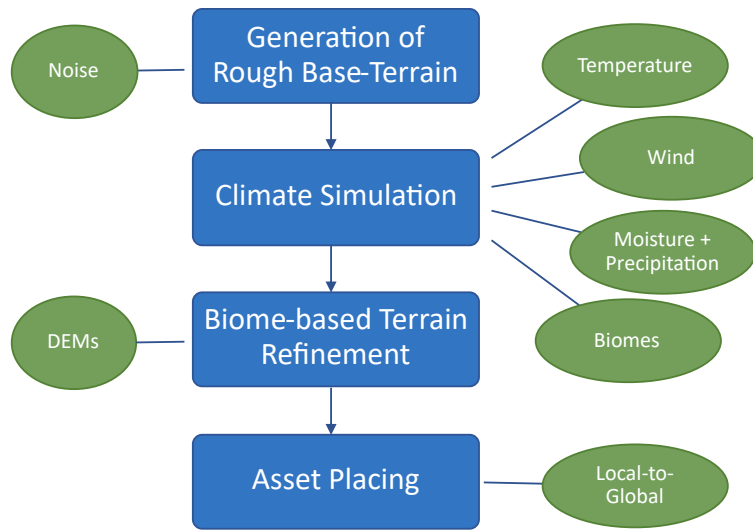


FIGURE 5.2: The concept of our terrain-generation system as a pipeline model.

The advantage of this approach is that we can use the different PTG styles in the individual pipeline steps and concatenate them in such a way that brings out the respective strength, which results in a better trade-off between the requirements. The synthetic noise functions are able to quickly generate a general terrain and are highly adaptable. The biome distribution is then computed using our physically-based climate simulation resulting in realistic-looking results while being easily adjustable by transparent parameters. Highly realistic biome-specific terrain features and details finally are quickly added by overlaying DEM images, which is an example-based approach. The individual steps of our pipeline and the chosen methods are described in more detail later.

For compatibility reasons with external applications, e.g., modeling tools or 3D rendering engines, we decided to represent our terrains by heightmaps instead of voxels. Additionally, we use different resolutions for the individual steps of our pipeline, for example, the climate simulation requires a less detailed grid (see Figure 5.3). The final heightmap can be exported as a set of tiles in a standard file format (grayscale images). It can also be used directly in the Unreal Engine 4 which enables us to use build-in techniques like LOD, instancing, and level streaming. In the following, we will detail the steps of our pipeline.

### Base Terrain

To generate the base terrain, we decided to employ synthetic PTG methods, specifically, noise functions. In this first step we only generate a rough terrain and such methods offer the most flexibility and widest range of possible terrains while also being very fast. Moreover, they are not limited in size or resolution. Physically- and example-based methods would be more restrictive, e.g., have more constraints between parameters or need specific example images, and the potential benefits of greater realism and more details are not relevant as we refine the terrain in later steps. The drawback of noise functions, the need for tedious fine-tuning to get

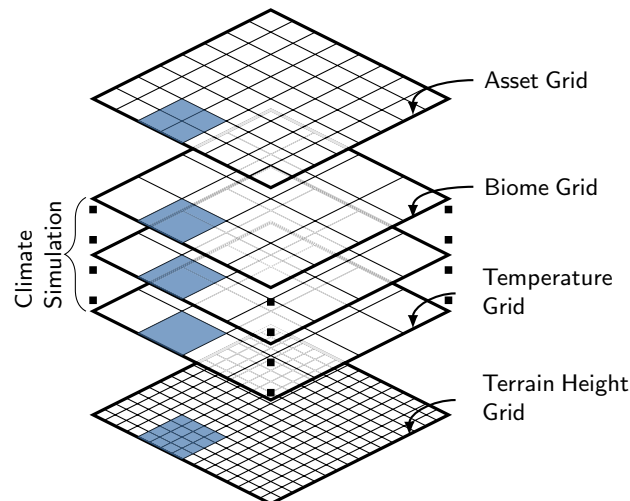


FIGURE 5.3: The used data structures as a stack of regular grids. The layers can have different resolutions.

realistic-looking results, does not apply because only the high-level terrain has to be generated.

We create the rough terrain by relying on common noise functions, more precisely, multiple octaves of simplex noise (using [276]) as this is well suited to generate a general fractal terrain. This method is fast, scalable, not too complex regarding usability, and sufficient as a coherent, coarse basis. The noise parameters, as well as the number of octaves, can be set by the user. However, replacing or adding other noise functions for more diverse base terrains would be an easy modification. A user-definable threshold marks the sea level to distinguish between land and water bodies (see Figure 5.4).

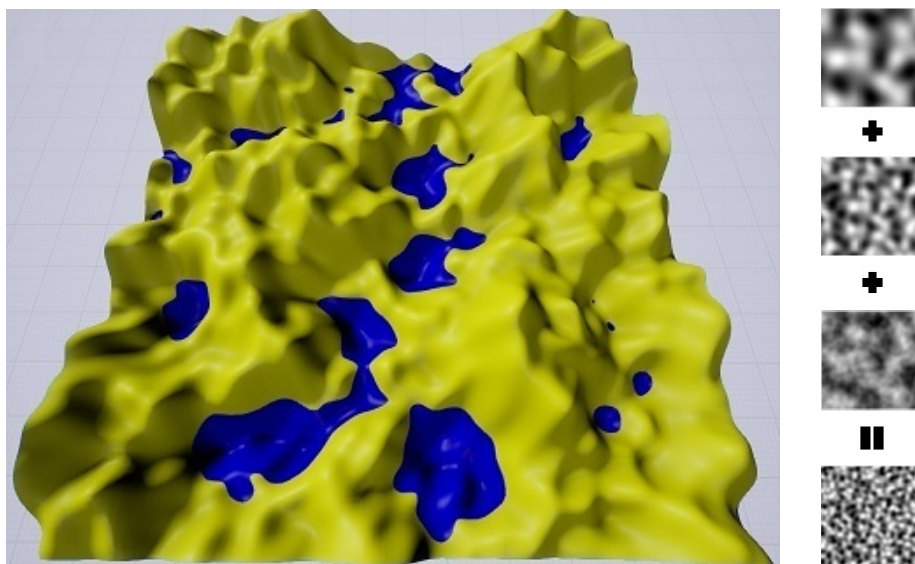


FIGURE 5.4: Left: Example terrain after the base terrain generation using fractal noise. Areas in blue represent water bodies. Right: Fractal noise as a combination of multiple octaves of simplex noise.



## Climate Simulation

For the next pipeline step, the computation of the biome distribution, we use a physics-based approach in contrast to other fully-synthetic methods that often rely on noise. We have developed a climate simulation that allows the generation of realistic, or at least plausible, distributions with a couple of easy-to-understand parameters. By comparison, noise functions would, in our mind, entail more fine-tuning or result in less realistic terrains, and sketch-based methods would require more manual work which we want to avoid. However, sophisticated simulations are more computationally expensive, which is why we disregard some effects to simplify the system and focus on reasonable approximations. The goal of our climate simulation is to add physically-plausible realism to the terrain while still being moderately fast to compute.

Following our pipeline-based design, the climate simulation is composed of multiple, sequential steps by itself, namely, temperature, wind, and precipitation computation, and lastly the biome classification. In detail, our climate simulation works as follows:

- The first step in our climate simulation is the temperature computation. Here, we provide two different interpolation methods: a bi-linear interpolation (see Fig. 5.5 (left)) and a sine-based alternative (see Fig. 5.5 (right)). The former provides more flexibility for the user, while the latter is more suited to model one-dimensional gradients resembling the behavior observable on the earth between the equator and the poles. Both modes account for a height-based temperature falloff to simulate the temperature decline which occurs with increasing height, and are easily adjustable with a few parameters.

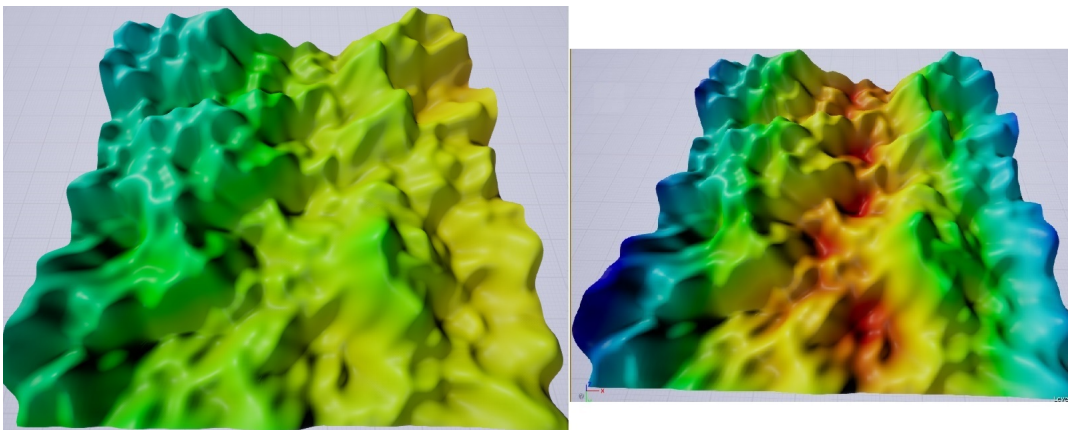


FIGURE 5.5: Example terrains after the temperature calculation. The temperature is depicted by the color: red meaning hot and blue meaning cold. On the left, the bi-linear corner interpolation was used, while on the right, the sine-based interpolation was employed.

- The next step is the simulation of the prevailing wind to distribute the later generated moisture over the terrain. In order to keep the performance reasonably high, we use an iterative approach to calculate the wind directions instead of applying a computationally expensive fluid dynamics solver. Our method is a simplified version of the semi-Lagrangian scheme [353]. We dropped the diffusion process and the pressure calculations as we handle these separately

in a later pipeline step. Therefore, we only consider external forces and self-advection to simulate the wind and compute its directions in a vector field. For these two components, we developed a less computationally expensive algorithm. Figure 5.6 shows an example.

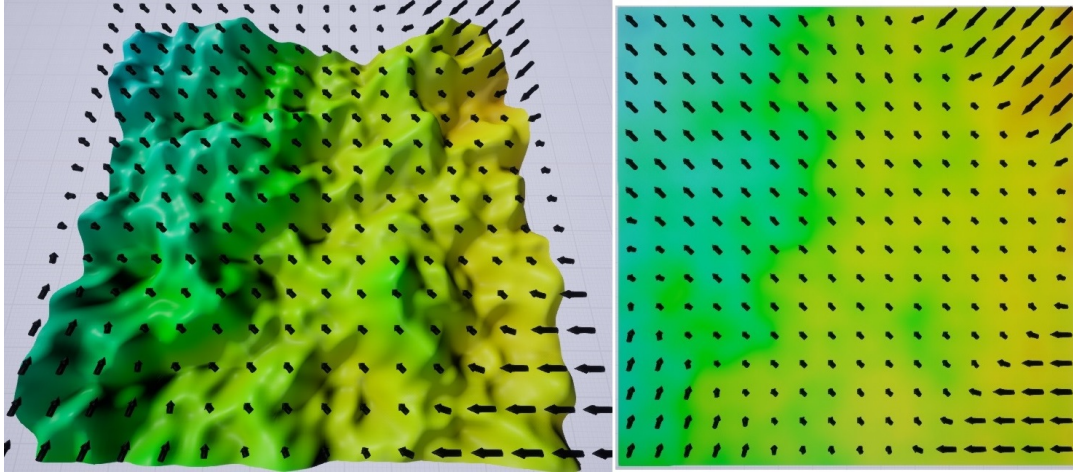


FIGURE 5.6: Example terrain after the wind calculation. The vector field of arrows depicts the computed prevailing winds. The right image shows a top-down view in which the terrain got flattened for a better overview.

The basic idea is to specify initial values for the four corners which act as the external forces. An iterative approach distributes the wind directions on a vector field: In each iteration, the new wind direction for each cell is computed by combining it with its adjacent cell in the wind direction and adding a little random deviation to simulate micro disturbances. Finally, we additionally consider the closest corner to model the persistence of the external forces. This delivers a plausible smoothing or cancellation behavior along the dynamically moving fringes between the main wind currents. Algorithm 4 depicts the wind calculation process.

- In the third step, we use the wind and temperature data to compute a precipitation distribution for the terrain. Again, we decided to use an iterative simulation-based approach. Basically, cells marked as water represent moisture sources. The evaporation is modeled as a temperature-dependent function; in fact, it can be chosen between an exponential and a linear version. The wind currents are responsible for distributing the moisture. Most of the moisture gets transported to the neighboring cell in the direction of the wind, but some shares also are transferred to the two cells adjacent to the neighbor and source. The actual distribution depends on the wind's direction and the previous moisture amount of all affected cells. With this algorithm, it is possible to model some form of dispersion and equalization. The amount of precipitation occurring depends on the local moisture and temperature and is modeled as a two-step process. First, the precipitation arising during moisture transport is computed. By using the previously computed temperature values and calculating the difference between the target and source, we can also simulate natural phenomena like rain shadows. Finally, additional precipitation is computed for moisture-holding cells to simulate other, more local causes. Again,

**Algorithm 4** Computation of the prevailing winds.

---

**Require:** user parameters (and corner wind vectors) set

**for all**  $C$  in  $Grid$  **do** ▷ cells in wind vector grid

$C_{temp} \leftarrow DetermineNearestCorner(C)$

$C.Vector, C.Corner \leftarrow C_{temp}.Vector$

$C.CornerDist \leftarrow CalcDist(C, C_{temp})$

**for all**  $Iterations$  **do**

$Grid_t \leftarrow []$  ▷ second, temporary wind grid

**for all**  $C_t$  in  $Grid_t$  **do**

$C \leftarrow GetCorrespondingCell(C_t)$  ▷ in main grid

$\alpha \leftarrow CalcAngleDeg(C.Vector)$

$C_{target} \leftarrow CalcTarget(C, \alpha)$

**if**  $C_{target}$  is valid **then**

$C_t.Vector \leftarrow AVG(C.Vector, C_{target}.Vector)$

$C_t.Vector \leftarrow C_t.Vector + CalcRndDeviation(C_t.Vector)$

$W \leftarrow CalcForceWeigth(C.CornerDist)$

$C_t.Vector \leftarrow w \cdot C_t.Vector + (1 - w) \cdot C.Corner$

$Grid \leftarrow Grid_t$

---

exponential or linear formulas can be used. Although we provide reasonable standard values, the system can be modified by a set of user parameters steering the formulas and therefore the results. For more details, see Algorithm 5. Figure 5.7 shows examples of a terrain and its moisture and precipitation distributions.

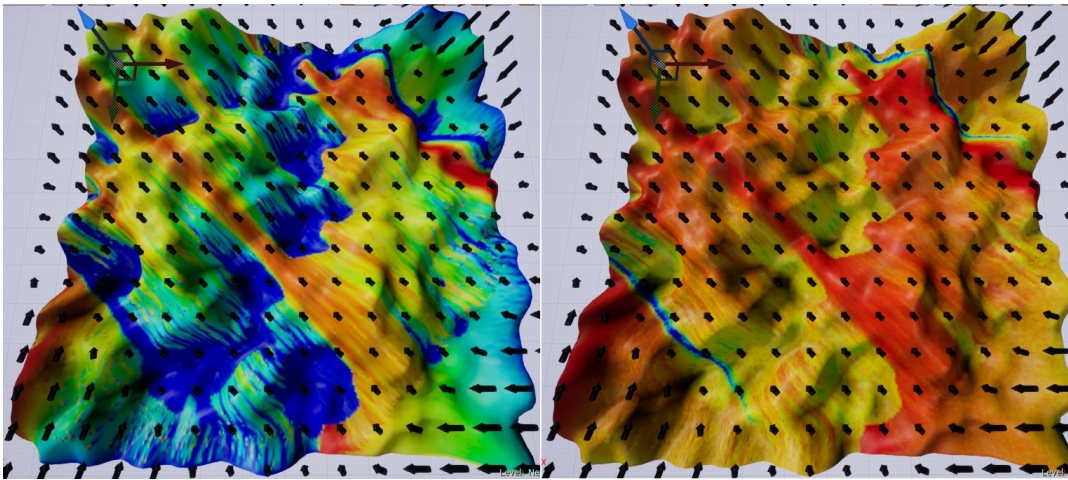


FIGURE 5.7: Example terrain after the precipitation calculation. The left image shows the moisture and the right image the resulting precipitation. Areas in red represent low values while blue areas represent high values of moisture and precipitation, respectively.

**Algorithm 5** Calculation of the precipitation.

---

**Require:** all user parameters set

```

for all  $C_m$  in  $Grid_m$  do                                ▷ all cells in moisture grid
   $Angle \leftarrow CalcAngle(WindVector(C_m))$ 
   $MP \leftarrow NewPacckage(C_m)$                             ▷ new moisture package with sender cell
   $CalcSetReceiver(MP, Angle)$                                ▷ stores  $MP$  in all receiving cells

for all Iterations do
  for all  $C_m$  in  $Grid_m$  do
    if Linear then                                       ▷ lin./exp. evaporation mode
       $Evap \leftarrow CalcLinear(C_m)$ 
    else
       $Evap \leftarrow CalcExp(C_m)$ 
    if  $C_m.Type()$  is WaterCell then
       $C_m.Gain \leftarrow C_m.Gain + Evap$ 
    else if  $C_m.Type()$  is Border and BorderMoisture set then
       $C_m.Gain \leftarrow C_m.Gain + Evap \cdot BorderFactor$ 
    for all  $MP$  in  $C_m.Packages$  do                        ▷ all receiving moisture packages
       $AVG \leftarrow CalcMoistureAVG(MP.Receivers, Grid_{mo})$ 
       $Share \leftarrow CalcShare(ExpansionFactor, AVG, RedistFactor, MP.Angle)$ 
       $M_{wind} \leftarrow Share \cdot MP.Sender.Moisture$ 
       $MoistureTransports.Pushback(M_{wind}, MP)$            ▷ (list) incoming moist.
       $C_m.Moisture \leftarrow C_m.Moisture + M_{wind}$ 
    if  $C_m.Moisture < MaxMoisture$  then
       $C_m.Moisture \leftarrow C_m.Moisture + C_m.Gain$        ▷ capped at  $MaxMoisture$ 
    if  $MoistureTransports$  not empty then
      for all  $MoistValue, MP$  in  $MoistureTransports$  do
         $T_{diff} \leftarrow Grid_{temp}[C_m] - Grid_{temp}[MP.Sender]$ 
        if Linear2 then                                   ▷ lin./exp. precipitation mode
           $Precip \leftarrow CalcLinPrecip(MoistValue, T_{diff})$ 
        else
           $Precip \leftarrow CalcExpPrecip(MoistValue, T_{diff})$ 
         $C_p.Precip \leftarrow C_p.Precip + Precip$ 
         $C_m.Moisture \leftarrow C_m.Moisture - Precip$ 
       $P_{rnd} \leftarrow CalcRandomPrecip(C_m)$ 
       $C_p.Precip \leftarrow C_p.Precip + P_{rnd}$ 
       $C_m.Moisture \leftarrow C_m.Moisture - P_{rnd}$ 
   $Grid_{mo} \leftarrow Grid_m$ 
   $Clear(Grid_m)$ 

```

---

- The last step of the climate simulation is the classification of the resulting biomes according to the computed properties, in particular, the temperature and precipitation. For this purpose, we use a slightly modified and discretized Whittaker diagram [410] as a lookup table. For each pair of temperature and precipitation values, a specific biome ID is assigned according to the lookup table. In principle, other classification systems are possible as the lookup table can be freely changed or replaced by the user. Figure 5.8 shows an example terrain after the biome classification.

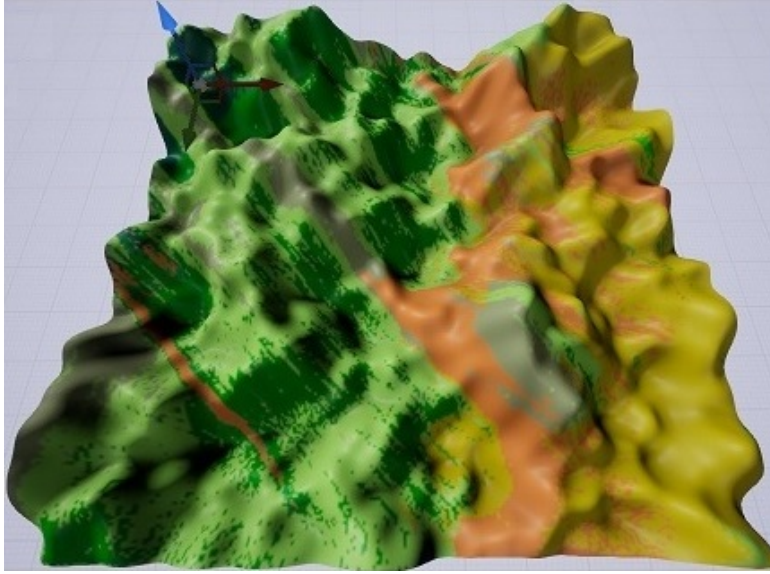


FIGURE 5.8: Example terrain after the biome classification. The colors denote different biomes, e.g., orange depicts a hot desert, and light green depicts grassland.

### Terrain Refinement

To complete the terrain generation, the rough base terrain is enriched with more realistic details based on the biome distribution provided by the climate simulation. We decided to use an example-based approach, in particular, DEMs, to obtain realistic biome-specific terrain details because of the vast pool of freely available DEM data which can be exploited. The DEMs serve as examples that can be blended onto the base terrain. The advantage is that the DEMs inherently provide realistic biome-specific terrain features and details. To get such realistic details, other methods would need a lot more user tuning and manual work, e.g., crafting specific multi-layered noise functions for each biome type, or complex computations in the case of physically-based methods. Figure 5.9 depicts the refinement process.

Another aspect that has to be considered for multi-biome terrains is, that especially organic, natural-looking biome transitions are essential. Therefore, we further customize the previously computed biome borders. Algorithm 6 details the process but the basic idea is to initially use user-adjustable, simplex-based fractal noise to distort the borders at a more granular level. For this purpose, we allocate a higher-resolution biome grid. Compared to more sophisticated techniques from the field of texture synthesis, this is a simple and fast-to-compute method that guarantees a result with consistent quality. Depending on the input data, other methods may occasionally result in technically correct but visually unsatisfactory results like straight biome borders (e.g., graph cut).

In a second step, we compute a biome-based DEM weighting using a convolution kernel to blend the adjacent biomes and their corresponding DEMs: Each DEM weight equals the area of the corresponding biome inside the kernel boundaries proportional to the whole kernel region. The strength of the resulting blend and the required computation time depend on the size of the kernel, which can be set by the user. The final DEM value can be easily calculated as a weighted sum over the occurring DEMs. For simplicity, we assume a one-to-one relationship between the DEM

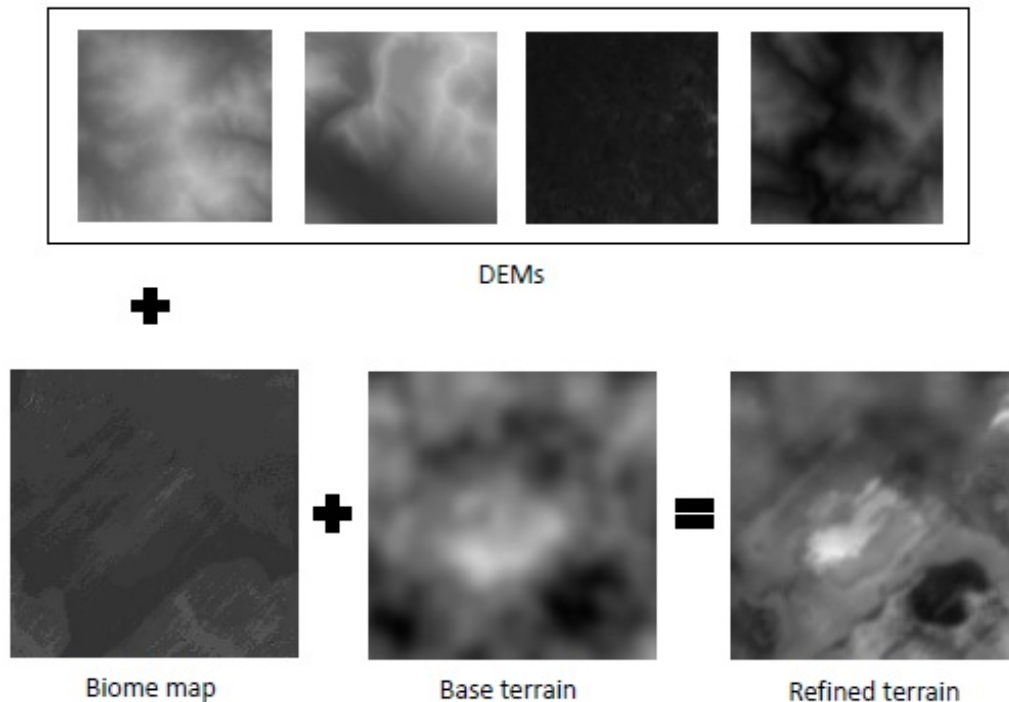


FIGURE 5.9: Biome-based terrain refinement: The base terrain is combined with biome-specific details that are obtained using DEMs.

texels and the terrain heightmap. Finally, we combine the generated biome-specific detail layer with the base terrain by using a weighted sum of the two heightmaps.

### Asset Placement

In the final step of our pipeline, we populate the biomes by placing assets. We have developed an iterative, rule-based local-to-global model, that, in contrast to global-to-local models, enables the creation of emergent distributions. Additional advantages are, that the model can easily be modified or extended by further constraints, and the individual assets, through the defined rules, inherently consider the biome transitions. We also considered using a global-to-local model in combination with real plant distribution data, but such data is hardly available for all kinds of biomes.

Our system is designed to use pre-modeled meshes, which allows for arbitrary generation methods to be used. However, the mesh generation itself, in a modeling sense, is not part of this work. We provide a basic database of pre-defined assets that can be easily extended by the user. Each asset is associated with a set of properties, e.g., clustering probability, shadow tolerance, or repelling distance. An example is depicted in Fig. 5.10. The placement is done iteratively via the dart-throwing principle where a random position is sampled and checked for the assets constraints, see Fig. 5.11. Our sampling approach is generally based on Poisson-disk sampling, where all the points are guaranteed to maintain minimal distances between each other. However, we extended this basic approach to cover also more complex multi-object distributions with bilateral constraints. Yet our approach is very flexible through the easy-to-understand parameters which steer the placement. We divide the assets into a few main classes, e.g., organic- and inorganic, with corresponding relevant parameters, which helps to improve the usability. Additionally, assets are partitioned into size categories which are processed iteratively such that

---

**Algorithm 6** Process of the terrain refinement.

---

**Require:** user parameters set, DEMs available

```

for all Biomes do
    DEMs[Biome] ← LoadRandomDEM(Biome)
    DEMs[Biome] ← ShiftMinHeight(0)
    BioMap ← CreateBiomeMap()
    NoiseMap ← CreateFractalNoise()
    for [horizontal, vertical] do                                ▷ one iteration each
        for all Cb in BioMap do                                ▷ all cells in biome map
            Cn ← GetNeighbour(up)                             ▷ right in second iteration
            if Cb.biome not equal Cn.biome then
                Offset ← NoiseMap(Cb)
                Biome ← BiomeMap(Cb + Offset)                ▷ in up/right dir.
                for all Co between Cb and Cb + Offset do
                    Co.Biome ← Biome
    for all Ch in Gridheight do                                ▷ cells in height grid
        BiomeCounters ← InitAll(0)                               ▷ list of counters
        Kernel ← CalculateKernel(KernelSize)
        for all Ck in Kernel do                                ▷ cells overlapped by kernel
            C ← C + 1                                           ▷ counter
            BiomeCounters[Ck.Biome] ← BiomeCounters[Ck.Biome] + 1
        for all BC, Bio in BiomeCounters do
            Fraction ← BC/C
            Ch.DEMHeight ← Ch.DEMHeight + Fraction · DEMs[Bio](Ch)
            Ch.Value ← Ch.DEMHeight · DEMFactor + Ch.BaseHeight · BaseFactor

```

---

smaller assets consider the previously placed bigger ones. With this technique, we achieve more plausible mixed distributions and environments. Generally, depending on the parameters, it is possible to model clustered, random, or uniform distribution and anything in between.

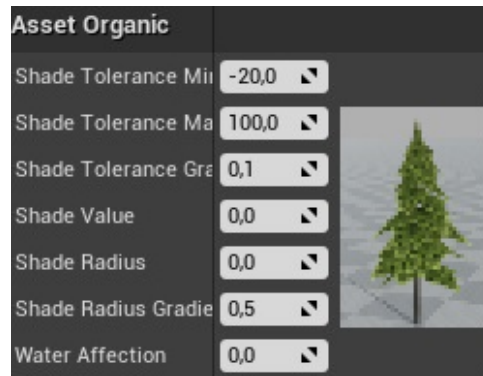


FIGURE 5.10: Example of an asset (tree) with various properties that steer the placement algorithm.

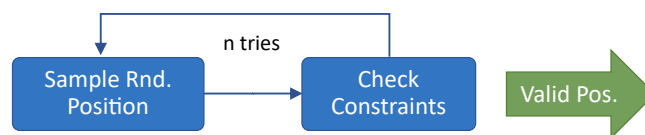


FIGURE 5.11: Process of finding valid asset spawn positions using dart throwing.

As seasons have a significant influence on the terrain cover's visual appearance, each asset can be associated with up to four different meshes representing its seasonal look. The meshes then are swapped automatically according to the current season, which can be changed in real-time. Additionally, the UE4 provides instancing which improves the rendering performance, and a LOD system for dynamic switching between the placed assets' detail levels.

#### 5.1.4 Results

We have implemented our terrain generation system directly in the Unreal Engine 4.20 using mainly C++ programming. It is directly accessible via the Unreal Editor which makes it very convenient for content creators.

We are not aware of quantitative measures to evaluate the quality of automatically generated terrains or biome distributions. Hence, we decided to provide mainly a qualitative evaluation of our system. We, e.g., show the influence of several of the most important parameters in the terrain generation pipeline. Additionally, we present measurements of the performance of each pipeline step in various configurations. Moreover, we are not aware of any directly comparable scientific work with the same focus – fast procedural multi-biome terrain.

First, we investigate the plausibility of the generated terrains. Figure 5.12 shows an example of terrain generated with our approach.

Different biomes can be easily distinguished by different surface characteristics. The distribution of the biomes is a result of correctly simulated natural phenomena like the occurrences of rain shadows. The easy accessibility of the adequate and meaningful parameters from the editor makes it easy to generate a vast variety of



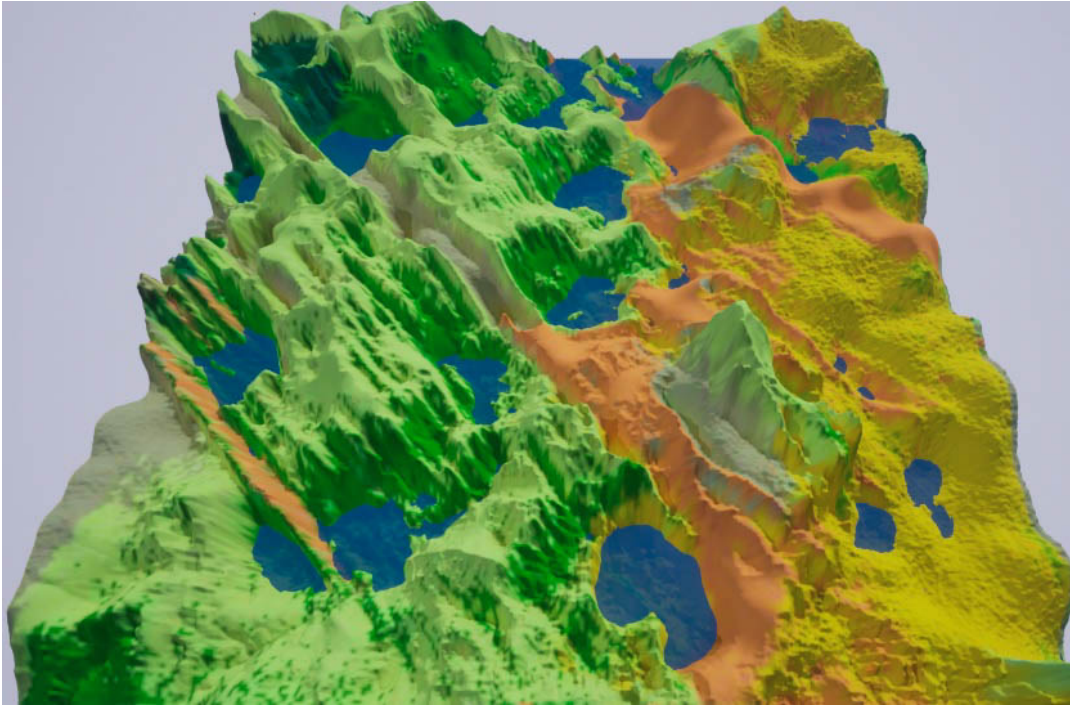


FIGURE 5.12: A final terrain of our PTG system rendered in the Unreal Engine (representing  $1600 \text{ km}^2$ ). The different surface characteristics caused by the distribution of multiple biomes can be seen easily. Each biome is also depicted by a different color.

different terrains and asset distributions. For instance, Figures 5.13a, 5.13b and 5.13c show the influence of the temperature, the wind direction, and the user-based noise, while all other parameters remained constant: the resulting terrains look very different, however, they are still plausible. Figures 5.13d, 5.13e and 5.13f show a variety of terrains if we change multiple parameters. The interaction of the different parameters like base noise, temperature, wind direction, and chosen DEMs is responsible for the even more wide range of generated terrains. These example terrains show not only diverse occurring biomes and biome distributions but also drastically varying surface characteristics and general patterns.

Placing hundreds of thousands of assets with cross-class dependencies to populate a huge terrain is easy with our system (see Figure 5.14, where the generated terrain is populated with around 200,000 assets).

By changing the asset-placement parameters, the user can directly influence the distribution and density of the assets while maintaining high realism (see the Figures 5.15a, 5.15b and 5.15c).

The real-time-adjustable season of the year has a significant impact on the landscape's appearance, as can be seen in Figure 5.16, and enables the visualization of an even wider range of environments and adds an additional layer of realism to the user if switched dynamically.

Additionally, we have investigated the performance of our terrain generator. All timings were done on a Windows 10 PC with an Intel Core i7-7800X processor, NVIDIA GeForce RTX 2070 graphics card, and 16 GB system memory. The running time of the pipeline depends mainly on the resolution of the particular grids. The expected running time is  $\mathcal{O}(n^3)$  with  $n$  denoting the number of cells per axis. This is dominated by the simulation of wind and precipitation with an expected running

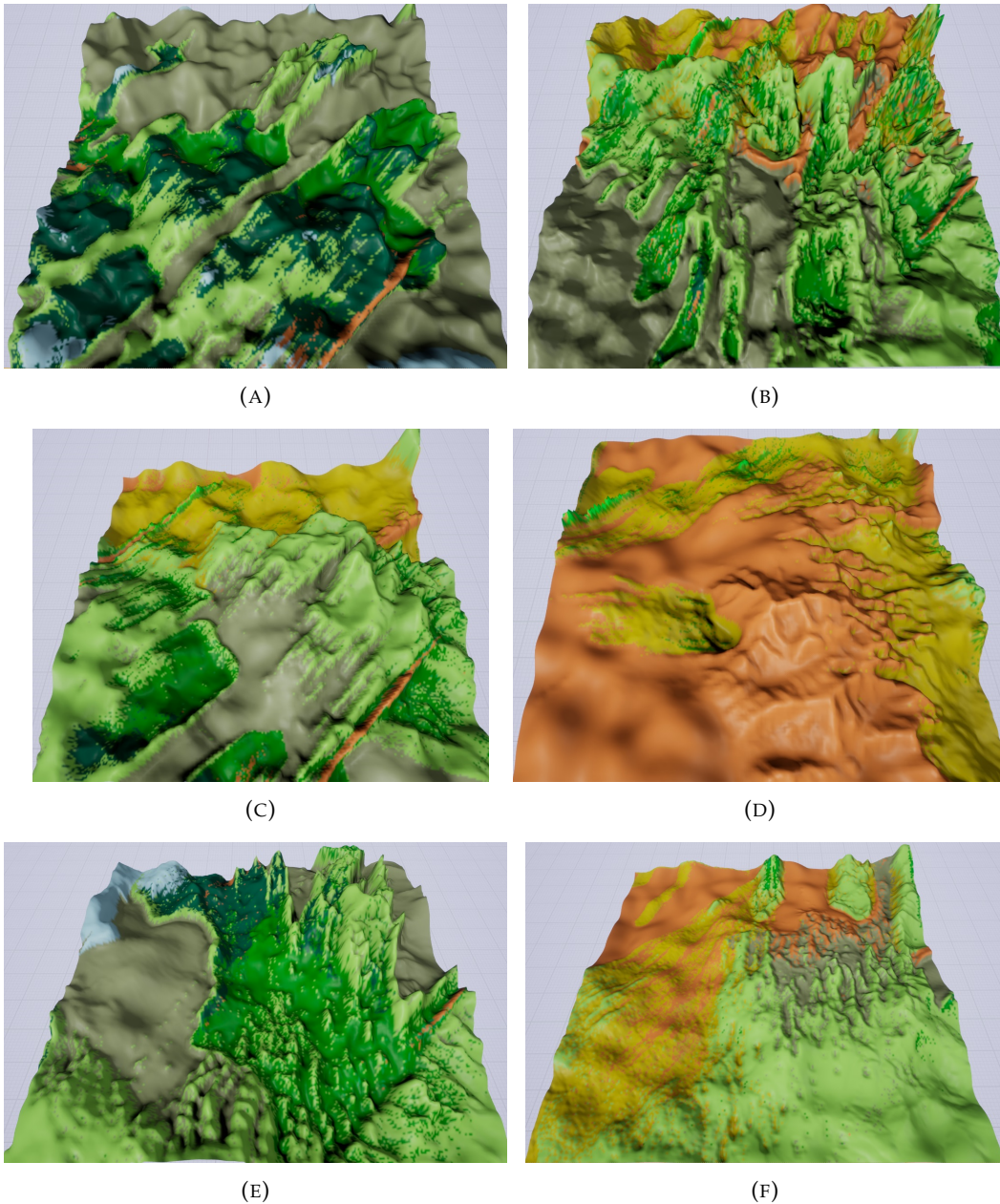


FIGURE 5.13: A set of different terrains generated by our system, visualized on a proxy mesh with biomes depicted by different colors. In contrast to Figure 5.8 we changed the temperature (a), the wind directions (b), the base noise (c) numerous parameters at once (d), (e), (f). Especially the last three examples show how a wide range of different terrains with wildly varying surface characteristics and biome distributions can be generated. This is a result of the interaction of parameters like the base noise, temperature, wind direction, and chosen DEMs.

time of  $\mathcal{O}(n^3)$ . The other steps are expected to have a running time of  $\mathcal{O}(n^2)$ . However, the biome classification is actually bound by the constant DEM loading times and the asset placement is nearly linear in the number of assets. The theoretical memory consumption is  $\mathcal{O}(n^2)$ . In practice, it is dominated by the number of asset instances.

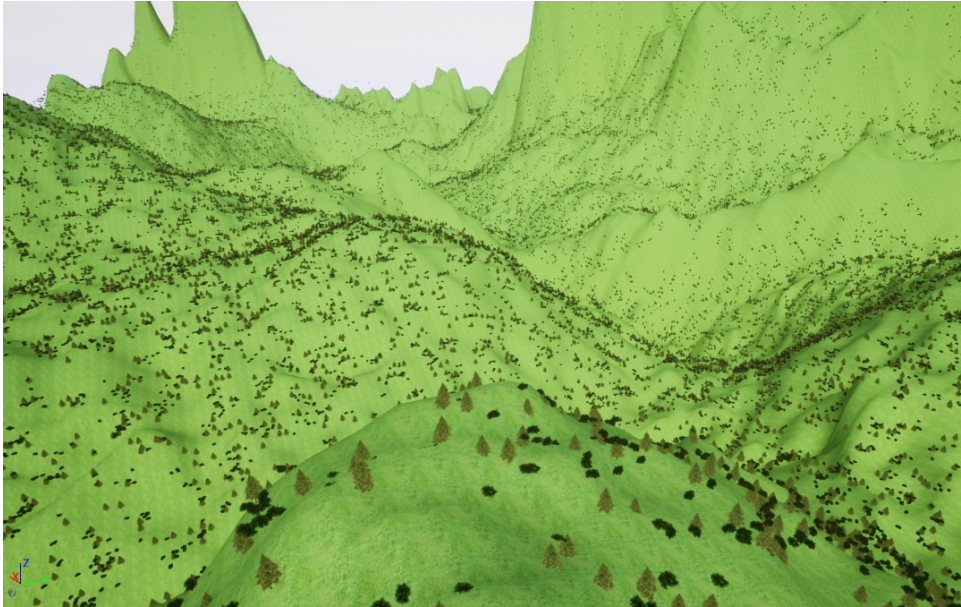


FIGURE 5.14: The same final terrain as in Figure 5.12 with procedurally distributed assets using our asset placement component. Around 200,000 instances were spawned in total.

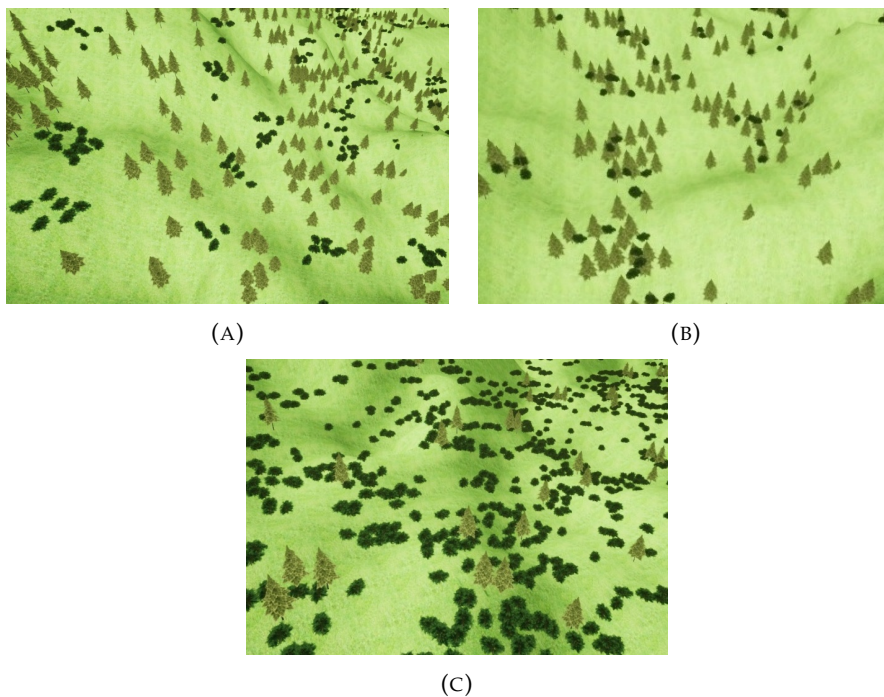


FIGURE 5.15: Three different asset distributions generated by our system. (a): Tight clusters of shrubs in open spaces between trees. (b): Shrubs growing exclusively in shadowed areas within dense tree clusters. (c): Dense, clumped distribution of shrubs around loosely grouped trees.

Figure 5.17 shows the time needed for calculating the individual pipeline steps with respect to the grid resolutions. All times were measured by performing several test runs using the Unreal Engine profiling tool and taking the median time over all

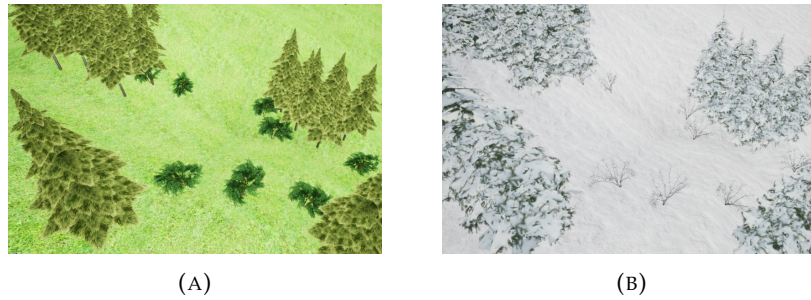


FIGURE 5.16: A terrain with its cover at different seasons: summer on the left (a), winter on the right (b). Seasons and the corresponding meshes can be automatically switched by our system.

test runs.

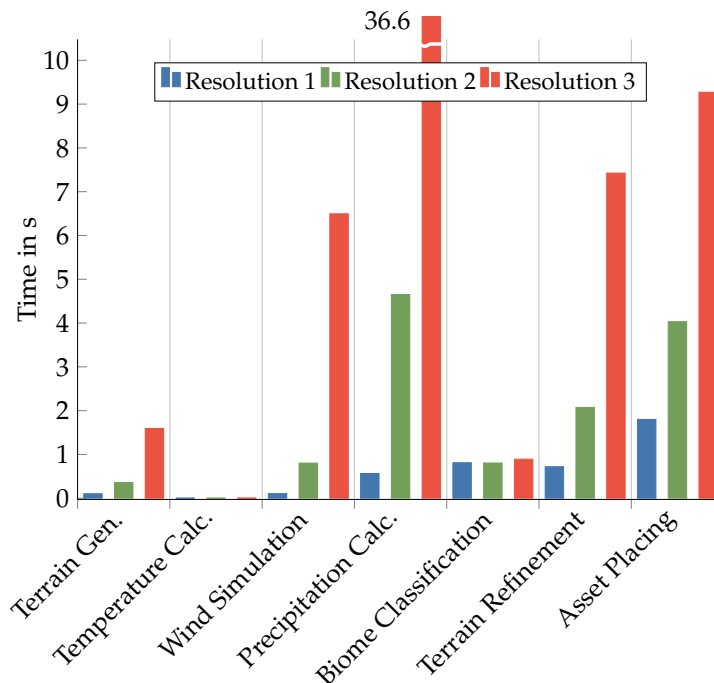


FIGURE 5.17: Computation times of the different pipeline steps: For the first pipeline step, the cell resolutions (per axis) were set to 1024, 2048, and 4096, respectively. For the distribution of assets, the resolutions were set to 30, 60, and 120 assets per cell, while for all other steps, the resolutions are 128, 256, and 512 cells per axis.

The computation time of the individual pipeline steps differs significantly, from a few milliseconds for the temperature calculation up to 36.6 seconds for the precipitation calculation in the most expensive configuration. However, the computations last in the majority of cases less than ten seconds. The overall most time-consuming steps are the precipitation calculation and terrain refinement, as expected, but also the asset placement. The main factor affecting the needed computation time is the respective grid resolution, but for some pipeline steps, additional parameters also have a great influence. For example, during the refinement of the terrain, the blending part takes the most amount of time and therefore, the adjustable blending kernel

size has a significant impact on the performance of this step. Figure 5.18 shows how different kernel sizes affect the performance for the terrain refinement.

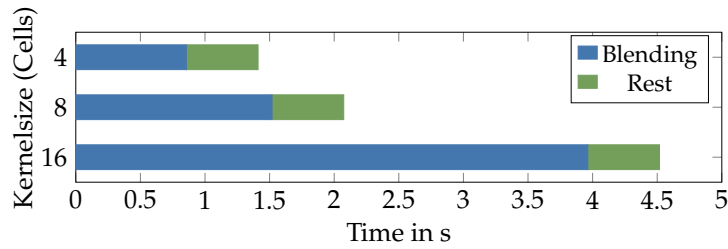


FIGURE 5.18: Computation time of the terrain refinement step with different sizes for the blending kernel (size per axis). The resolution was fixed to 2048 cells per axis. The middle bar corresponds to the terrain-refinement result with resolution two in Figure 5.17.

In the case of distributing the assets, it is noteworthy that the computation time is highly dependent on the combination of the strictness of the placement rules and the maximum number of iterations per individual placement. Our tests show that roughly 100 - 300 assets can be placed in a couple of seconds. The time to spawn the assets in Unreal is included in the placement time and, in general, takes up a considerable amount of it, which is a bit surprising. Figure 5.19 illustrates the interaction between these parameters and the composition between the time needed for the positioning and the instance spawning. A higher number of positional tries leads to both, more instances being actually placed as well as a higher computational time, as expected. However, with stricter placement rules which are harder to fulfill, i.e., increasing the minimum distance between instances, the overall time can actually decrease even if the time needed for calculating the positions increases relatively. This is due to fewer valid positions being found and therefore fewer instances being spawned which is a fairly costly process.

However, even for large grid sizes, our terrain generation requires less than a minute in almost all cases. These fairly low computation times meet our expectations and enable quick iterations. Implementing multi-threading could improve the performance even further as most algorithms are prone to parallelization.

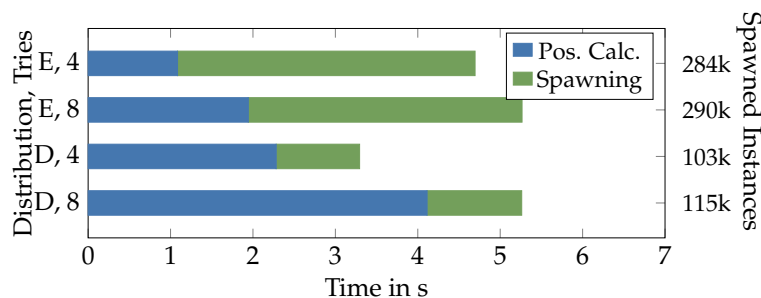


FIGURE 5.19: Calculation time of the asset placement step, partitioned in the phases of position calculation and actual spawning in the UE, with different distribution rules and a varying amount of positioning tries per instance. D stands for a difficult to fulfill rule-set and E for an easy one. Also, the resulting number of spawned assets is stated; The optimal amount would have been 300k, where k stands for thousand.

### 5.1.5 Conclusions and Future Work

We have presented a pipeline-based system for procedurally generating multi-biome landscapes. Our pipeline model is easy to use and flexible on both, local and global scale. Our system can help level designers and other users with generating and quickly iterating over vast and yet visually plausible multi-biome terrains. This process includes the automatic but still user-adjustable population of the terrain with huge amounts of pre-defined assets following complex distributions. Utilizing a carefully simplified climate simulation was a central element in the success. It is not only crucial for creating the biomes themselves and their realistic distribution but is also the basis for other landscape aspects, e.g., our DEM-based terrain refinement and also the asset placement relies on the specific biomes. Our results demonstrated that the generation is reasonably fast while the terrains are visually plausible.

The modular pipeline approach is an excellent base for future work. There are many possibilities to extend our system or improve existing parts even further. The most promising additions, in our opinion, would be to implement a simulation step for river and erosion generation and the introduction of different geological layers and soil types. Also encasing the system in a meta-iteration to alternate between biome distribution and terrain generation could be interesting for producing results that are even more realistic. Regarding improvements, multi-threading and a more complex wind simulation would be the most important ones. Finally, we see much potential in investigating the use of neural networks, e.g., style transfer or GANs, for terrain generation, specifically, generating DEMs and combining them with the base terrain.

## 5.2 Procedural Generation of Landscapes with Water Bodies Using Artificial Drainage Basins

After we previously presented a PTG system that is able to generate landscapes with various biomes, in this section, we want to tackle an often overlooked aspect of procedural terrain generation, namely, generating landscapes that feature plausible water bodies. Therefore, we propose a method for the procedural generation of huge landscapes that focuses on creating realistically-looking river networks and lakes as well as a natural-looking integration. We achieve this by an approach inverse to the usual way: we first generate rivers and lakes based on artificial drainage basins and then create the actual terrain by “growing” it, starting at the water bodies. Our pipeline approach not only enables quick iterations and direct visualization of intermediate results but also balances user control and automation. That means the first stages provide great control over the layout of the landscape while the later stages take care of the details with a high degree of automation. Our evaluation shows that vast landscapes can be created in under half a minute. Also, with our system, it is quite easy to create landscapes closely resembling real-world examples, highlighting its capability to create realistic-looking landscapes. Our implementation is easy to extend, highly compatible with external applications thanks to using heightmaps as underlying data structures, and thus, can be integrated smoothly into existing workflows.

The work presented in this section is based on our published paper PC1 in Appendix A.

### 5.2.1 Introduction

As described in Section 5.1, procedural generation of 3D landscapes is a research topic of great relevance, as the interest in large, realistic digital landscapes is steadily rising throughout many fields and industries. We established, that such digital landscapes get employed, for example, in computer games, but also movies, simulators, and virtual testbeds, and that producing realistic and detailed terrains while keeping the workload in check is always a challenge. Naturally, there has always to be a trade-off between control and automation as fully procedural generated landscapes usually do not meet specific requirements but designing everything by hand is not viable either. With greater and denser worlds the challenge is only getting more exacerbated. Numerous works have been presented on the automatic generation of landscapes and terrains using procedural generation techniques, including our own work that we presented in Section 5.1.

A sub-topic that got much less attention despite being highly relevant for large landscapes is the procedural generation and plausible integration of water bodies, i.e., mainly rivers and lakes but also other features such as smaller streams and ponds. While rivers/river networks, their natural processes, and interaction with the surrounding terrain have been – and still are – extensively studied in related fields such as geology, ecology, and hydrology [308, 46, 47], relatively few works focused on procedural generation of 3D representations of them in near real-time speed. Existing scientific models and simulations usually employ only 2D representations, are more focused on analyzing existing landscapes than creating novel ones, or are very time-consuming to perform. For instance, one popular model to create river networks is optimal channel networks (OCNs) [301, 14]. Brown et al. [38] gives a good overview of the different works and approaches to create digital rivers throughout the various research fields.

We propose a method and pipeline for quick and easy procedural generation of large, plausible-looking landscapes which include and integrate believable water bodies, i.e., river networks. In our approach, we mimic the mutual influence between terrain and water bodies by first generating the rivers and lakes based on artificial drainage basins, and then computing the final terrain. This way, we get more natural-looking landscapes, than by retroactively adding rivers to a terrain. In order to demonstrate our proposed approach, we have developed a prototype application in Unity. In this prototype, we have applied a pipeline approach that makes it easy to evaluate intermediate results and emphasizes a workflow with quick iterations. Finally, we have conducted an extensive evaluation of our proposed system.

### 5.2.2 Related Work

One of the oldest approaches for procedural terrain generation is to use subdivision techniques such as the midpoint displacement and the diamond square algorithms, and noise functions (e.g., simplex and ridge noise), as they are able to produce fractal-like structures, which are also often found in nature. Moreover, such techniques are, generally, relatively easy to use, highly scalable, and computed quickly. A comprehensive overview of various noise functions is given by Lagae et al. [197]. The drawbacks of those techniques are the intrinsic lack of control over global features, and the un-intuitive parameters, which make it hard to create geologically plausible landscapes.

A popular approach to providing intuitive control to the user is to add an authoring phase at the beginning, most often in form of a user sketch that acts as a high-level constraint for the subsequent terrain generation [369, 116].

An approach to create more realistic terrains is to mimic or simulate natural processes. For instance, Michel et al. [249] create folded terrains with mountain ranges by using simplified plate tectonics that is based on user sketches and Cortial et al. [76] approximate the movement and collision of user-authored tectonic plates on a planetary scale. Simulation-based techniques most often focus on thermal or hydraulic erosion, the latter normally encompassing some form of fluid simulation. One example is Mei et al. [244], who used an adapted shallow-water model to calculate the erosion and deposition process as well as the sediment transport. The proposed work was implemented on the GPU. Stava et al. [355] further improved the method and combined two hydraulic erosion algorithms. Cordonnier et al. [75] combined tectonic uplift from user-provided input maps and simulation of hydraulic/fluvial erosion based on the stream power equation to generate plausible large landscapes. However, despite efforts to speed up the computations, most simulation-based approaches are very time-consuming, especially if applied on large-scale terrain. Another disadvantage is the lack of intuitive control over the generated terrains.

Realistic-looking large-scale terrains also can be created using example-based procedural generation techniques such as texture synthesis. For instance, Zhou et al. [442] employ patch-based texture synthesis to generate terrains based on user sketches and example DEMs. Gain et al. [117] instead switched to parallel pixel-based terrain synthesis for higher efficiency; user control is provided by several modifiable, local constraints. Guerin et al. [135] presented an example-based authoring pipeline in which the user provides a quick sketch of the main terrain features, and then a set of neural-network-based terrain synthesizers creates the corresponding terrain. The synthesizers – which are conditional generative adversarial networks – get trained on real-world example data. Naturally, example-based methods are limited by the available example data and can only replicate terrain features and landforms that are represented in the input DEM. Also, high-level geological constraints and the correct relations between large-scale features such as drainage basins are usually not taken into account.

Relatively few works explicitly focus on procedural rivers and water bodies as initial terrain-defining elements, although river networks play an important role in the natural formation of the terrain. Kelly et al. [180] were the first to propose the idea of procedurally generating terrain based on river networks and corresponding drainage basins. Here, the river networks were generated based on constrained midpoint displacement, and then the terrain was computed accordingly. Derzapf et al. [82] employed a similar approach but applied it on a planetary scale. In the work by Teoh [373], the terrain generation starts, too, by first procedurally creating river networks. In this case, rivers are grown from randomly placed outlets around the land region. In contrast to these works, Genevaux et al. [127] explicitly take hydrological knowledge into account, additionally, initial user sketches provide more control. Based on the sketch, a river-network graph is created, river segments get classified into different types of watercourses, and the surrounding terrain gets computed using a hierarchical terrain construction tree. Zhang et al. [436] present a similar approach but generate the rivers based on Tokunaga river networks and calculate the surrounding terrain using a diffusion process.

For a more comprehensive overview and discussion of procedural terrain generation techniques, we refer to the presented work by Galin et al. [119].



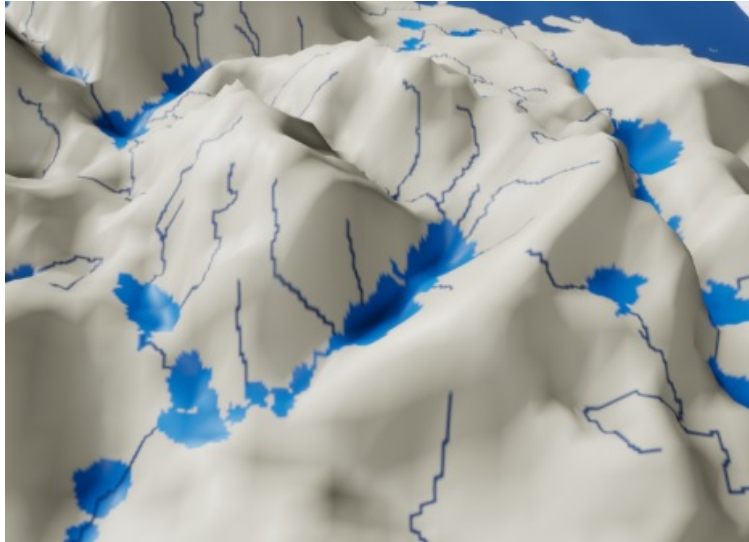


FIGURE 5.20: Computing river networks using A\* pathfinding. The approach is quick and is able to directly generate lakes too, but the results are not convincing.

### 5.2.3 Overview of our Approach

In this section, we will present an overview of our proposed methods and pipeline. First of all, we will briefly discuss different approaches to procedural terrain generation that include and emphasize river systems, and explain the reasoning behind our approach.

The first group to consider is purely simulation-based approaches. These are able to produce realistically-looking terrains with river networks, e.g., using erosion simulation. As we prioritize a quick and easy generation over absolute realism, though, we have decided against these simulation-based approaches. Another approach that follows the classical order of first generating the terrain and then adding rivers to it is to use pathfinding algorithms. For instance, an adapted A\* pathfinding algorithm can be used to follow the terrain downwards from randomly placed sources. This not only has the advantage of being computed rather quickly but also that lakes can be easily computed by just defining the searched area as a lake, which tends to be in local minima. However, in our experience, the river networks and their integration into the terrain were not convincing, as they did not respect geomorphological constraints. Figure 5.20 shows an example of employing this approach for river generation. In this thesis, we instead propose to follow the more natural “rivers first” approach, specifically, first generating river networks based on artificial drainage basins and then modeling the final terrain after them. Accordingly, we will present several methods and an integrated pipeline to allow for that.

For the implementation of our pipeline, we have used heightmaps as data structures for data and terrain representation, as they have a smaller memory footprint and, most importantly, provide much greater compatibility with external applications than voxels. Even though we do not aim for real-time generation (i.e., an online algorithm), it is very important to facilitate a quick workflow for the users, which means having fast computation times as well as direct visualization of intermediate results that can be made modifications upon. These considerations lead us to employ a pipeline approach in which each step should be computed in a matter

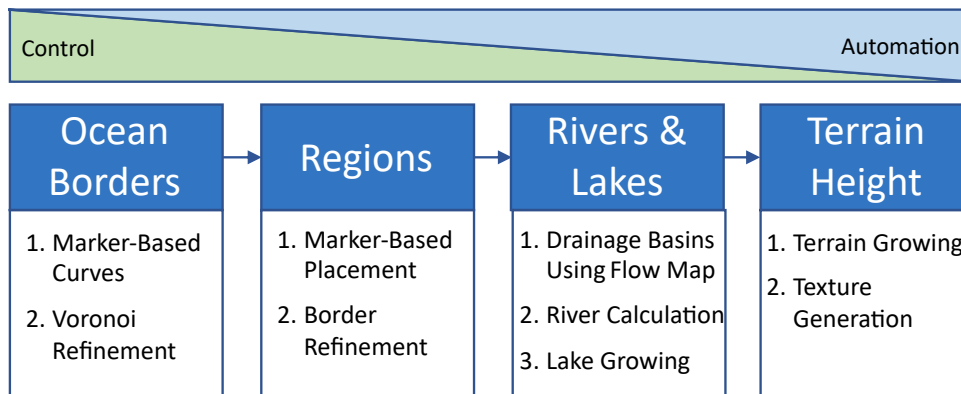


FIGURE 5.21: Overview of our procedural generation pipeline. The first stages feature more user control, the latter ones provide more automation.

of seconds, be repeatable if modifications are desired, and the results directly be applied on a proxy mesh for inspection. Fig. 5.21 depicts a high-level overview of our approach. We start with the general landscape layout by letting the user author the landmasses using marker-based curves. Then, different regions can be marked (e.g., flatland, or mountains). Following this, the river networks, including lakes, are computed based on artificial drainage basins. Finally, the terrain height gets computed based on the previous steps. In the earlier stages, we emphasize providing the user with more control over the algorithm, while the later stages have a higher degree of automation. The reasoning for this is that the user should have a great influence over the general layout and shape of the landscape and its landmasses, which are defined in the earlier stages of the pipeline, but not be overwhelmed with a host of detail decisions all over the landscape. Smaller modifications at the places where it is deemed necessary could be better done afterward via a polishing pass with a sculpting tool.

## 5.2.4 Our Terrain Generation Pipeline in Detail

### Ocean Borders

Our pipeline begins with the coastlines that separate the landmasses from the ocean. As the user should have great flexibility to shape the general terrain to his needs, we prioritized providing great manual control instead of excessive levels of automation for this part of the pipeline. To define the overall shape of a landmass, the user can place marker objects throughout the scene, which we then compute a closed curve from. This curve defines the coastline: the area within forms the landmass, and the rest represents the ocean (or the other way round, depending on a user setting), see Fig. 5.22. Multiple landmasses can be built by repeating the process. If no markers are placed, the whole scene forms a single landmass without an ocean. The markers contain location information, a rotation, and a strength parameter affecting the marker's influence, specifically, the curvature of the tangent (higher strength values resulting in less curvature). The curve is then computed by interpreting the markers as knots/control points of a cubic spline, the evaluation is done in the order the markers were placed. The final segmentation into land and ocean is stored in a regular grid whose granularity can be set by the user.

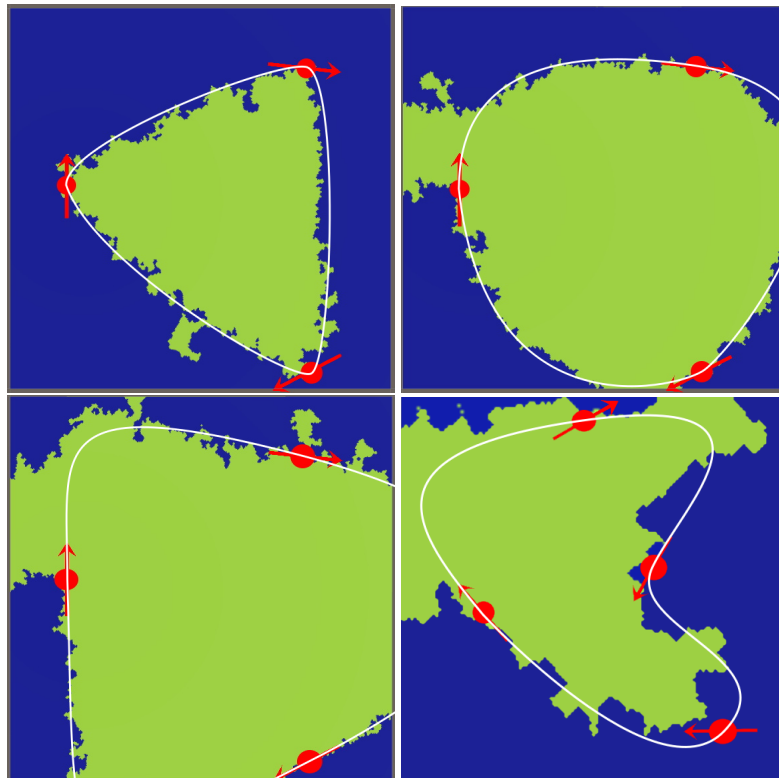


FIGURE 5.22: Authoring of the general shape of the landmass borders using marker objects which act as spline control points (red points with arrows). The white curve represents the generated spline and the green area depicts the final refined landmass. First 3 images: increasing strength values (reducing the curvature of the tangent).

In order to improve the outlines and get a more natural look, we have developed multiple refinement options for constrained randomization based on Voronoi diagrams. Generally, we randomly place Voronoi sites on the map and define the regions using the Manhattan distance. We use this metric, as the computations are done on a regular grid; other metrics can be applied, too, though. In the first refinement variant, each Voronoi region gets classified to represent landmass or ocean, depending on if the corresponding Voronoi site is inside or outside the spline, see the top left image in Fig. 5.23. This ensures that the general layout defined by the spline is retained, but the actual border is randomized. The granularity of the Voronoi diagram can be set by the user. To efficiently produce finer, more detailed borders, this refinement step can be followed up by a second iteration in which additional Voronoi sites are placed around the previously computed border, see the top right image in Fig. 5.23. Alternatively, this second iteration can be applied using a priority queue based on (fractal Perlin) noise for the growth of the regions, see Algorithm 7 for details of this process. Although computationally more expensive, it produces more varied results through more inhomogeneous region growth and can lead to the formation of small islands. The noise parameters influence the output, e.g., a higher frequency produces finer structures, and more octaves enable features of different scales. Fig. 5.23 (bottom row) shows an island with the different refinement variants applied.

---

**Algorithm 7** Border refinement (2. iteration, Voronoi + noise)
 

---

**Require:** 1. refinement iteration done, *RegionMap* set

```

B ← BorderCells                                ▷ From 1. iteration
Q ← []                                           ▷ Empty priority queue
for all VoronoiSites do
  P ← Random(B) + RandomXY(MaxRadius)
  Q.Enqueue(P, Noise(P))
B ← []
while Q not empty do
  C ← Q.Dequeue()
  for all N do                                ▷ Valid, unsearched neighbor cells
    RegionMap[N] ← RegionMap[C]
    Searched[N] ← True
    Q.Enqueue(N, Noise(N))
  if CheckIfBorder(C) then
    B.Add(C)
  
```

---

## Regions

After the landmasses are defined, the next step is to partition them into regions of different terrain types. Terrain types we have implemented are flatland (default), mountain, and desert. Others can be added easily, though. One option to divide the landmass into different regions could have been to use cellular noise. However, in this pipeline step, we again focused on giving the user great control over the layout of the regions by allowing the manual distribution of regions over the terrain. Alternatively, a random distribution is available too. A region is defined similarly to the coastlines in the previous step by placing region markers. This time they contain parameters regarding the terrain type, the extent, and its border, which is randomized

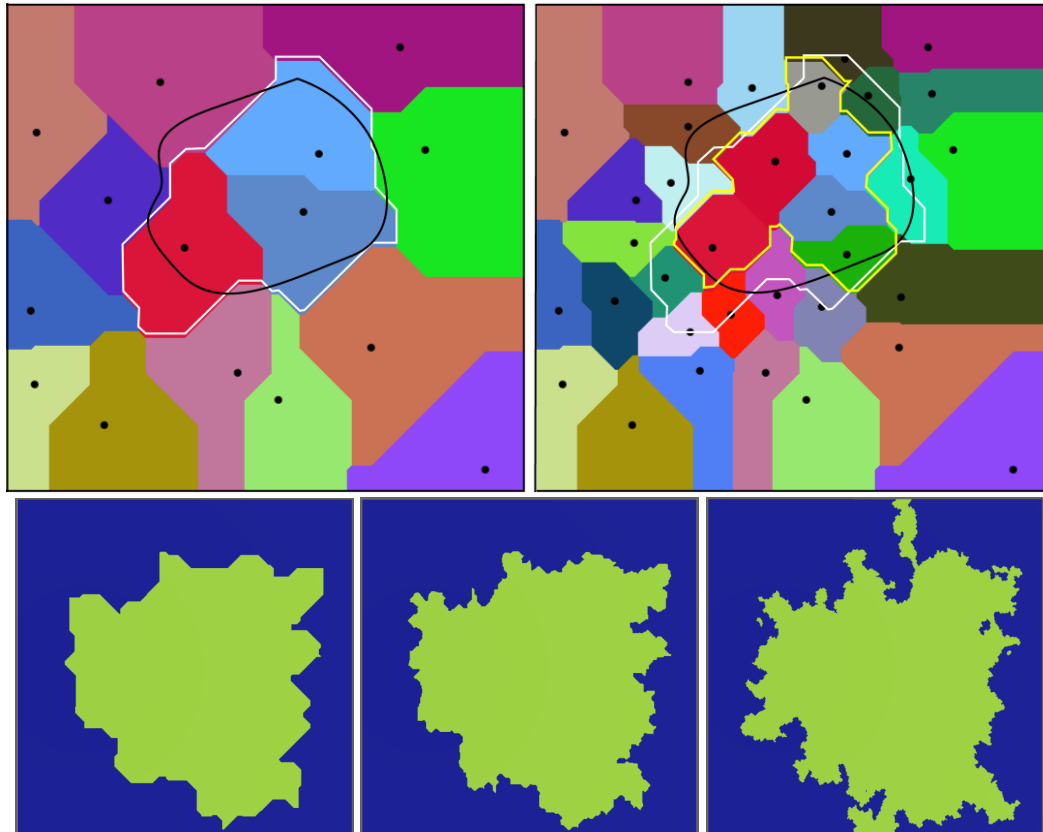


FIGURE 5.23: Border refinement based on Voronoi diagrams. Top row: the two iterations of border refinement. The left image shows the result after the first iteration (the black ellipse depicts the initial spline, and the white outline is the ocean border). The right image shows the second refinement iteration with the final border (yellow outline). Bottom row: The left image shows the land after one iteration, the middle image after two iterations, and the right image after two iterations with noise-based sampling.

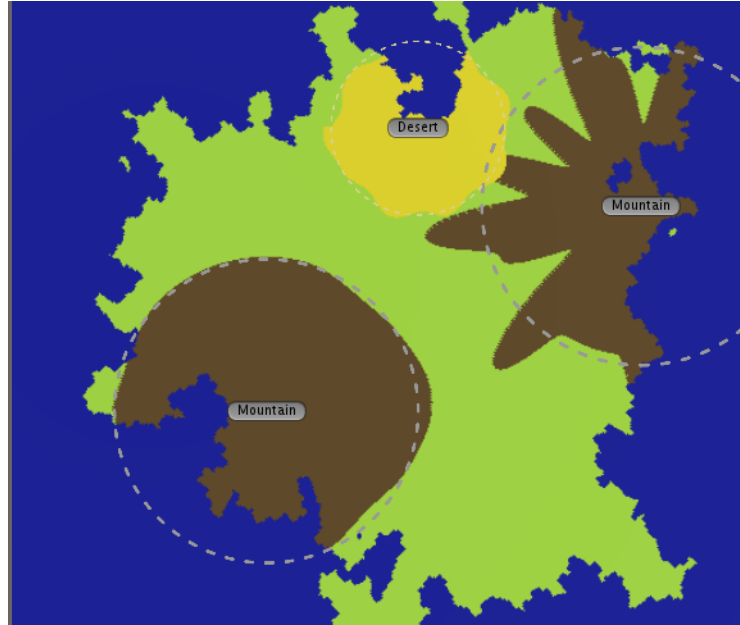


FIGURE 5.24: Segmentation of the land into different regions (green: flatland, yellow: desert, brown: mountains). Region borders can be customized using noise (note the different characteristics of the borders).

using noise. Fig. 5.24 illustrated an example terrain with three regions and different border characteristics. Region overlaps are solved according to the placement order. After the regions are placed, we iterate through all cells that represent land and store the id of the region they lie in.

### River Networks and Lakes

The next step in the pipeline is the generation of the river network. This is one of the most important steps in the whole pipeline, as it directly impacts the eventual terrain/heightmap generation. When analyzing naturally generated terrain, the landscape can be divided into catchment areas – also called drainage basins. These are the areas where water is collected by surface runoff (e.g., from precipitation). They are often divided by mountains or hills. This means that by artificially generating those catchment areas, we know afterward where we can place hills and mountains. To generate the catchment areas, we have developed a method inspired by optimal channel networks [301, 14]. Similarly to them, we define a finite graph  $G(V, E)$  over a regular grid spanning the landscape. The nodes  $v \in V$  correspond to the cells of the grid, and the edges  $e \in E$  link neighboring nodes and enable the flow of water between them. We then construct a spanning forest  $F$  over  $G$  that acts as a flow graph/flow network, i.e.,  $G$  gets partitioned into a number of spanning trees  $T$  – acyclic, directed, rooted sub-graphs. Each outlet acts as a root of one of these trees, which each represent one river network or its catchment area.

Our procedure is shown in Fig. 5.25 and starts by placing down several potential outlets around the coasts (left image) and also around mountain regions. The separation between those outlets happens so that the rivers can be generated differently based on terrain type. The number of outlets can be set by the user separately for

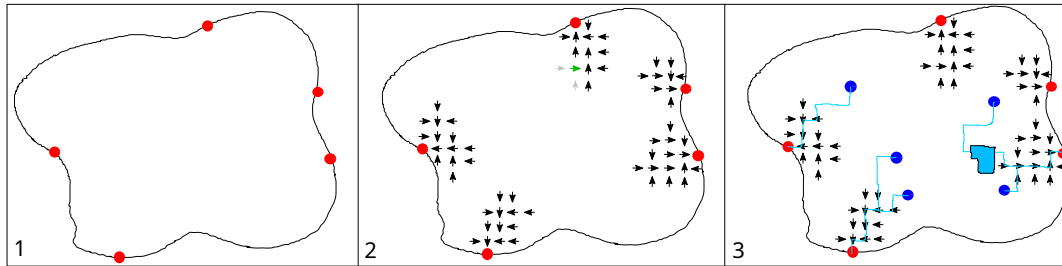


FIGURE 5.25: The process of computing rivers and lakes. First (1), random outlets are selected at the ocean borders (red dots). Then (2), we compute the flow directions across the land (see arrows), starting at the outlets and then consecutively and randomly selecting previously evaluated cells (green arrow in the highlighted area) to process their unprocessed neighbors (grey arrows). Eventually (3), we randomly place river sources (blue dots) and create the actual rivers by following the flow map (blue lines). Lakes are grown from random points on the river.

each region type. Placing outlets around desert regions is excluded from this process. Then, we compute the catchment areas by construction of the spanning forest that indicates the flow directions. This is done by calculating random flow directions for all cells in the uniform grid and storing them in a flow map (middle image of Fig. 5.25) that procedurally connects all cells from mountain and flatland regions. This process starts at the river outlets by adding all outlet cells into a fringe set. Then, consecutively, random cells from this set get selected and their unsearched, valid neighbor cells get processed by assigning the flow direction (pointing to the current cell) and, in turn, adding them to the fringe set. By using a random selection order, we guarantee a homogeneous growth of the spanning trees/propagation of the flow map and, thus, the drainage basins. Algorithm 8 shows the procedure in more detail.

This process also avoids desert regions to ensure that they stay dry when placing the rivers. The number of outlets influences the shape of the river networks. The fewer outlets are generated compared to the number of river sources, the higher the branching factor in the final river networks will be. This is because the individual rivers, starting at their sources, are more often routed to the same outlet and, thus, join somewhere along the way. Important to note is, however, that not all outlets will necessarily have rivers run into them. In Fig. 5.26 (left), an example landscape is shown with arrows depicting the corresponding flow map (due to the perspective there are slight offsets).

Following these preparations, we can proceed to generate the actual rivers. A possible approach for this would be to calculate the amount of water that would flow through each cell. This could be done by placing water evenly on the grid and following along each water unit with the previously generated flow directions while adding up how many water units traverse each cell. Rivers could then be placed in every cell that has an amount of water higher than a specified threshold. This method would work but lead to a relatively even distribution of river sources. Instead, we place the river sources randomly on the map and then follow the flow directions until the ocean is reached (right image of the image sequence of Fig. 5.25). This yields the advantage of a more random distribution of sources. The amount of river sources placed is again controlled by the user and can be set separately for

---

**Algorithm 8** River Generation

---

```

Q ← DistributeFlatlandOutlets(NumFO)    ▷ Queue w. outlets at ocean borders
while Q not empty do
  cell P ← Q.Pop()
  N ← RandomUnconnectedNeighbour(P, FlowDirectionMap)
  FlowDirectionMap(N) ← P
  HeightMap(N) ← HeightMap(P) + CalcGrowthFactor(N)
  Q.Enqueue(N)

Q ← DistributeMountainOutlets(NumMO)    ▷ At borders to flatland/ocean
for all O do                                ▷ Mountain outlets
  if O next to flatland then
    N ← RandomUnconnectedFlatlandNeighbour(O, FlowDirectionMap)
    FlowDirectionMap(N) ← O
    HeightMap(O) ← HeightMap(N) + CalcGrowthFactor(N)
  else
    HeightMap(O) ← 0

while Q not empty do
  cell P ← Q.Pop()
  N ← RandomUnconnectedNeighbour(P, FlowDirectionMap)
  FlowDirectionMap(N) ← P
  HeightMap(N) ← HeightMap(P) + GrowthFactor
  Q.Enqueue(N)

DistributeMountainSources(NumMS)
DistributeFlatlandSources(NumFS)
for all S do                                ▷ All flatland/mountain sources
  S.SetDry(Chance)
  cell P ← S
  while P.Type() not ocean do
    set type of P as (dried) river
    RiverStrengthMap(P) ← RiverStrengthMap(P) + RiverGrowthFactor
    P ← FlowDirectionMap(P)
    if NumLakes < MaxLakes and random chance then
      GenerateLake(P, RiverStrengthMap(P))

for all S do
  WidenRiver(S, FlowdirectionMap, RiverMaxWidth)

```

---



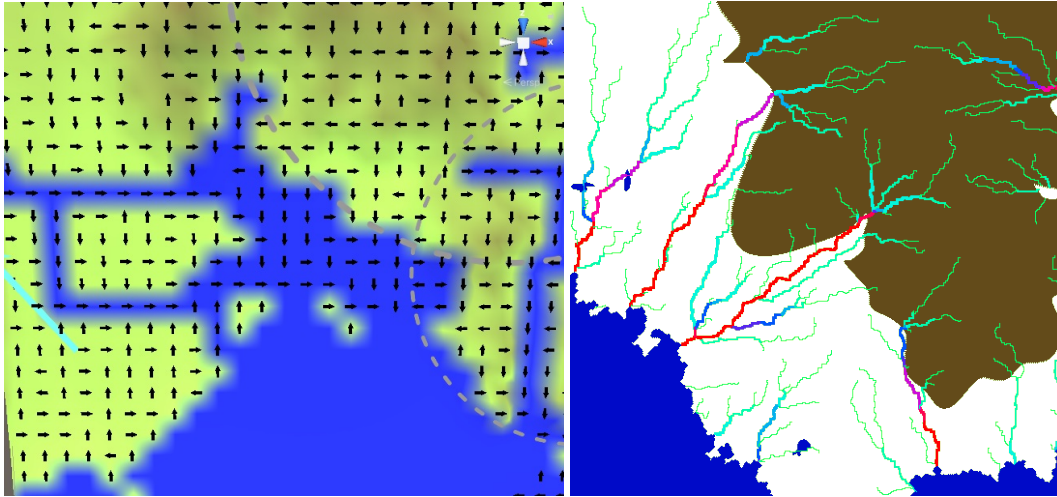


FIGURE 5.26: Left: An example map with arrows roughly indicating the corresponding flow map. Right: A map highlighting the river strength/width with different colors (green indicating weak rivers and red strong ones).

mountain and flatland regions. An example terrain with multiple rivers is illustrated in Fig. 5.27 (left). Users also can set the chance for a river to be a dried-out riverbed instead of a normal river. In our implementation, each river adds a certain amount of strength to a cell when flowing through it, thus, when two rivers join, their respective strength is combined from thereon. Based on this strength, we calculate the width of the river and accordingly assign more cells to it. Fig. 5.26 (right) depicts a color-coded example of the various rivers and their respective strength (red: strong, green: weak). The calculation of the width can be skipped by a user setting, though. It is also possible to set a maximum width for rivers. This can be useful when using a calculation grid with a lower resolution where one cell already covers a larger area of the final map.

For each cell a river travels through, there is a chance to generate a lake. The size of the lake depends on the strength of the river cell it is generated from: the higher the strength, the more likely it becomes for the lake to be bigger. In addition, the size is limited by user-set maxima and minima. To produce lakes that vary in shape, a noise function is applied to the lake borders. The user can set the frequency, the strength, and the number of octaves of the noise function. All the cells inside the border are classified as lake. In order to avoid overfilling the map with too many lakes and to give control to the user, it is possible to set a maximum amount of lakes generated for mountain and flatland regions. In addition, the lake generation is limited to one lake per river source. After generating a lake, the river generation continues. Fig. 5.27 (right) shows an example terrain with lakes and river networks in which the rivers have different strengths.

With our approach to create river networks, we are also able to generate distinct river deltas. For this, we perform a second, locally-bound iteration of generating drainage basins and then rivers. However, here we reverse the process: If a river delta should be generated, we select a river cell instead of a coastline cell as the starting point from which we compute the flow map; in this case only a regional one. Then, we select multiple coastline cells to be outlets from where rivers – the eventual distributaries of the delta – are created by following the computed regional

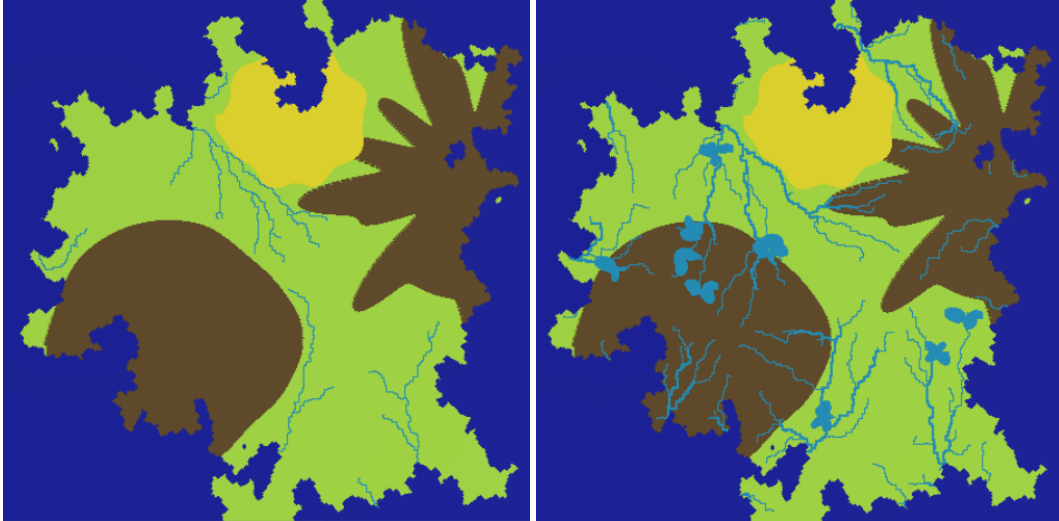


FIGURE 5.27: Left: A terrain with rivers generated in flatland (green). Right: The same terrain as in the left picture but with rivers of varying width and the addition of lakes.

flow directions back to the original river cell. We allow this process to randomly occur in the general vicinity of the ocean, see Fig. 5.28 for examples of this process.

### Terrain

After generating the rivers and lakes, we calculate the terrain’s height based on them and all the information that was produced in the previous steps of the pipeline. Algorithm 9 shows the process.

---

#### Algorithm 9 Heightmap Generation

---

```

Q ← all coast, river, lake cells           ▷ priority queue with height as priority
while Q not empty do
  cell P ← Q.Pop()
  for all N do                               ▷ neighbours of P
    if N unchecked and N.Type() not ocean then
      GrowthFactor ← CalcGrowthFactor(N)
      HeightMap(N) ← HeightMap(P) + GrowthFactor
      Q.Push(N)
      N.Checked(True)
  
```

---

The terrain will be “grown” starting at the oceans, rivers, and lakes and continuously rises while departing from them, see Fig. 5.29. To do this, all starting cells are added to a priority queue with the priority being the initial height of the cells. The initial height of water cells was computed during the river generation in the previous stage: cells marked as river directly get assigned a height which steadily increases from the outlet onwards based on a region-based steepness, although a distance-based curve is possible too. When a cell is removed from the queue, all the neighbors that have not been traversed yet will receive a new height by the addition of a growth value. The growth value is calculated by taking into account the terrain growth factor of the region the cell is in, and a random value obtained from

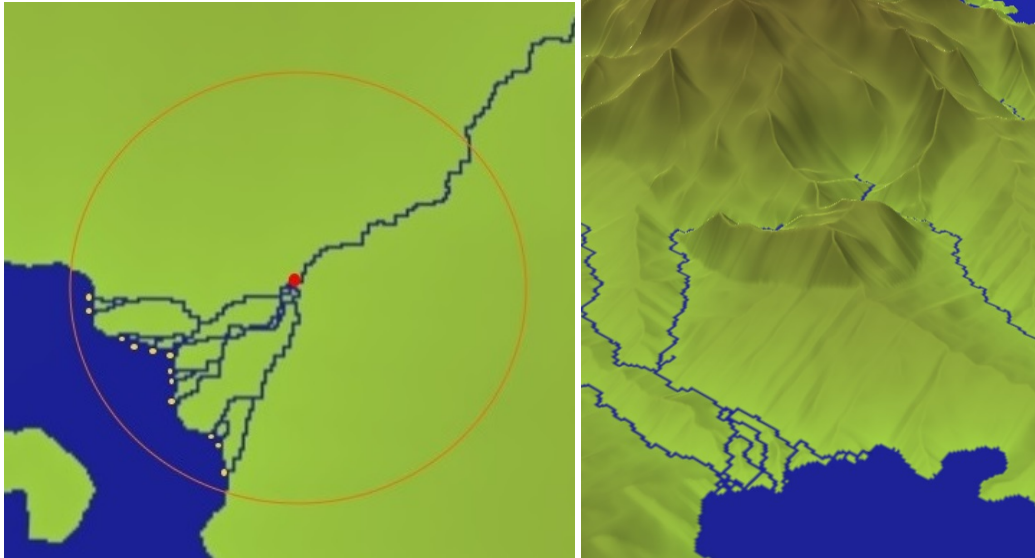


FIGURE 5.28: Two example landscapes in which river deltas were computed (using different visualization modes). The red dot in the left image depicts the position from which the river delta computation was started, the circle shows the radius of the local flow map, and the yellow dots mark the stream outlets.

noise functions that also depend on the region. The height value is also interpolated between the different region types to allow for a smoother transition between them. The exact height calculation for a cell  $x'$  which is the neighbor of an already computed cell  $x$  is described by the following equation:

$$h(x') = h_x + s_t \cdot b_d + s_r \cdot (1 - b_d) + |n_t \cdot a_t| \quad (5.1)$$

In this,  $h_x$  is the height of cell  $x$ ,  $s_t$  is the terrain/region-dependent growth factor which itself is computed using a distance-based interpolation between all regions in the area,  $b_d$  is a distance-based blending factor,  $s_r$  a growth factor for rivers,  $n_t$  the noise output whose frequency is again terrain-dependent, and  $a_t$  is a terrain-dependent amplitude. The latter is calculated as  $a_t = s_t \cdot k_t$  in which  $k_t$  is a terrain-dependent noise strength.

Fig. 5.30 shows two example terrains with computed height; note the more pronounced hills in the right image. The growth value itself does not change the final height of the terrain, though, as it is difficult to know exactly what the result would be. To guarantee the final height is controllable, the heightmap is scaled to fit the global world width and height parameters that can be set by the user.

### Visualization

The last step of the pipeline is the visual representation of the generated terrain. This is not part of the actual terrain generation but serves an important role in giving the user information about the results of the previous steps which allows for quick changes if desired. To visualize the terrain, we generate a mesh by first assigning a vertex to each cell of the heightmap and then calculating the corresponding triangles and UV coordinates. In default mode, the vertices' height is directly taken from the heightmap, but we also support a mode in which the mesh omits the height and

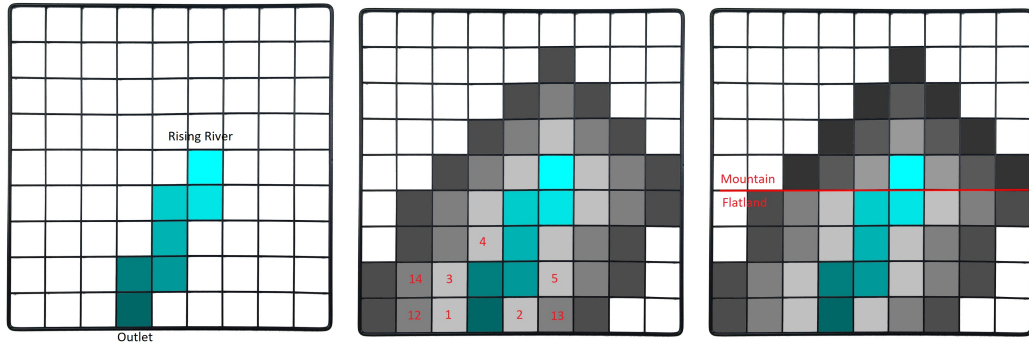


FIGURE 5.29: Terrain growing process. The height of river cells increases (brighter color) while departing from the outlet (left image). Then, the land is grown (darker color means higher) from the rivers outwards using a height-based priority-queue (middle image). The growth depends on the region (right image).

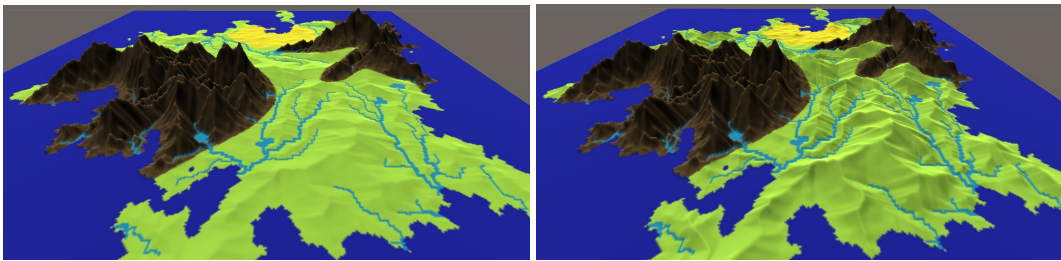


FIGURE 5.30: Final terrains with computed height. Note that increasing the slope parameter for the flatland (green) leads to higher, more pronounced hills (right image).

stays completely flat. This mode can be useful for earlier pipeline steps, e.g., the border generation, where the height information is not relevant yet, and a flat map is easier to evaluate. Finally, we generate textures for the terrain mesh. The user can choose from multiple rendering modes using different textures, which are illustrated in Fig. 5.31: The normal-texture mode displays a gradient that is dependent on the height, the region mode visualizes each region with a separate color, and the height mode displays a normalized heightmap as texture.

### 5.2.5 Results

In order to evaluate our proposed procedural system, we present a brief complexity analysis, extensive practical performance measurements, and a qualitative evaluation.

#### Complexity Analysis

The overall run-time complexity of our pipeline is

$$O(n_c \cdot n_s \cdot n_l) \quad (5.2)$$

with  $n_c$  being the number of grid cells,  $n_s$  being the number of river sources, and  $n_l$  being the number of lakes. This means that the time complexity depends linearly

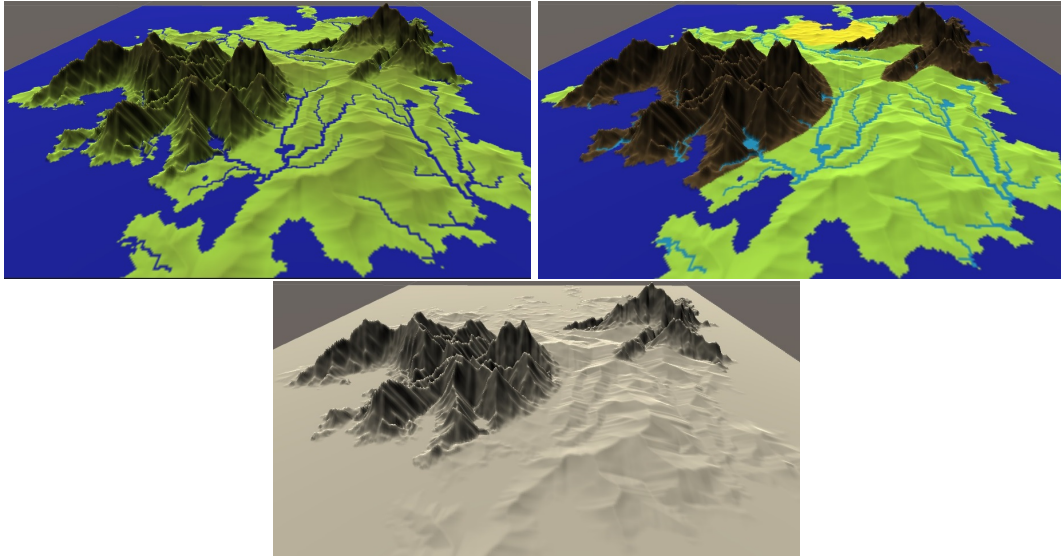


FIGURE 5.31: Different rendering modes highlight different aspects of the terrain. On the top left is the normal-texture mode with water bodies depicted in blue and a height-based color gradient for land. In the top right is the region mode, in which each region gets a different color, and rivers and lakes get colored in light blue. The bottom image shows the height mode with a black-and-white normalized height-based gradient.

on the number of river sources, the number of lakes, and the number of cells in the grid. Similarly, the space complexity is  $O(n_c)$ .

### Performance Evaluation

After the theoretical considerations, we did extensive real-world performance measurements of our system as a whole as well as of each pipeline step individually. All performance measurements were done using a PC with Windows 10, an Intel i7 7800x processor, 16 GB of main memory, and an Nvidia GeForce 2070 graphics card. As the performance is mainly dependent on the number of grid cells  $n_c$ , we conducted all measurements with sizes of  $n_c = 512^2, 1024^2, 2048^2$  and took the median of 20 runs.

In Fig. 5.32 we illustrate the computational time of the whole pipeline over the different grid sizes, whereby the timings of the individual pipeline steps are stacked on top of each other. As we can see, the computation is very fast: with low to

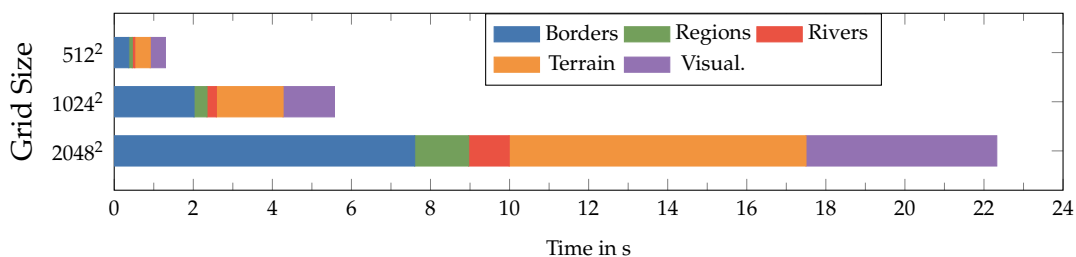


FIGURE 5.32: The calculation times for the complete pipeline over multiple grid sizes. Even with a grid size of  $2048^2$  the whole pipeline gets computed in under 25 seconds.

medium-sized grids, the whole process is done in a couple of seconds, single steps being computed nearly in an instant, and even with the biggest tested grid resolution of  $2048^2$ , the pipeline gets computed in under 25 seconds. Looking at the timings for the individual pipeline steps, the calculation of the region and rivers is the fastest and, compared to the other steps, negligible throughout all resolutions. At lower resolutions of  $512^2$ , the calculation of the terrain's height and the visualization take the most time with 0.39 and 0.38 seconds, respectively. At higher resolutions of  $2048^2$ , however, the border computation takes the longest with 7.6 seconds, followed by the terrain with 7.5 seconds. The reason for this is that the computational time of most pipeline steps grows with factors closely around the expected one of 4 that corresponds to the linear growth regarding the number of grid cells (quadratic regarding grid side length) which we established in the theoretical complexity analysis. The time for the border calculation, in practice, growth with factors around 5, though.

Investigating deeper what exactly causes the computational time in the individual steps, we find that in the river step, the calculation of the drainage basins via the flow map takes up  $> 88\%$  of the time while the following computation of rivers and lakes takes nearly no time, see Table 5.1. Accordingly, the number of rivers and lakes does not have a significant effect. The main factor regarding the performance is therefore the overall grid resolution. Some other parameters, however, also have a notable influence on the needed time for computation. For instance, in the first pipeline step – the border calculation –, the second iteration of border refinement takes considerably longer than the first one, as a higher number of Voronoi points is used. Also, if the second refinement iteration is applied with additional noise (default setting), it takes even more time to compute, as in this case, we use a priority queue. This is also the reason for the higher practical growth factor of this step. Regarding the visualization step, the computation of the textures takes roughly four-fifths of the time of the step while the mesh generation itself only takes one-fifth.

TABLE 5.1: Detailed timings of some pipeline steps for a grid resolution of  $2048^2$ .

Step	Substep	Time (ms)
Borders	Refine. (1i)	851
	Refine. (2i)	1601
	Refine. (2i+N)	7600
Rivers	Drainage B.	796
	Rivers+Lakes	100
Visual.	Mesh	800
	Textures	3426

### Qualitative Evaluation

To our knowledge, there are no metrics to quantify the quality and realism of procedural terrains and water bodies. Thus, in order to evaluate the quality and plausibility of the terrain generated with the proposed system and to showcase its versatility, we did a qualitative evaluation. For this, we have created several landscapes with different settings, which can be seen in Fig. 5.33, and reviewed the production process as well as the results regarding usability, flexibility, and plausibility. With our system, it is possible to produce a vast variety of shapes for the coastline. Single continents, as well as island groups, can be created by varying the number and position of land markers. Generally, we found the process very efficient and flexible;

simpler shapes can be realized very quickly with just a few markers, but by using a greater number, the user also can create more complex worlds. Similarly, the whole process to create a terrain is very easy and straightforward, as our pipeline design allows for quick iterations and the saving of intermediate results. Also, although we provide many parameters to fine-tune each step to the user's liking, in most cases the majority of them don't necessarily have to be changed and our pre-configured default settings will suffice.

Furthermore, several different height profiles can be generated. It is possible to generate large-scale maps, such as the examples in Fig. 5.33 but also landscapes at smaller scales, as shown in figure Fig. 5.34. Thanks to our focus on water bodies and the approach to create river networks before the final terrain, the procedural landscapes produced with our system are quite natural looking and feature plausibly embedded river networks that recreate the typical dendric structures from the real ones. In general, we found that the generated results look very plausible and, presumably, the majority of different demands on the produced landscapes can be satisfied.

To further evaluate the plausibility of the generated terrains, we have compared them with parts of the real world's terrain based on publicly available height data. For this comparison, we took DEMs – which represent elevation data of the real world's terrain –, constructed 3D meshes of them, and attempted to replicate the real terrain as closely as possible with our system while only investing a reasonable amount of time (a couple of minutes). As an example, we randomly took a section of the Severo-Evensky District in Magadan Oblast in Russia (61.21703, 160.21836) as a real-world reference. Fig. 5.35 shows the comparison between the mesh representations of both landscapes, the left image shows the real terrain, and the right one our systems replication of it.

As can be seen, it is possible to recreate a similar general shape of the coastline. Because the generation is heavily based on random components, it is impossible to generate a coastline that matches exactly. The mountain ranges are distributed with a good approximation of the reality. We were not able to acquire real-world references with satisfactory information about river networks, therefore, the map was generated only using dried riverbeds (no lakes, no explicitly visualized rivers). It is not possible to perfectly match the behavior where terrain touches the world border as the real map is a part of a larger landscape and thus, rivers flow through the border. Our terrain generation algorithm does not have information about the terrain outside the grid borders and thus cannot replicate this behavior. However, the inland parts of the river networks were generated in a believable way. Even though the pathways of the riverbeds differ from the original directions, the individual parts of river networks have similar overall shapes. This can be observed, for instance, in the northern parts of the mountains in Fig. 5.35. The heightmaps of both terrains are depicted in Fig. 5.36. Again, the general shape, as well as the dendric structures caused by the rivers, resemble the original, although they do not match perfectly. In general, it was possible to create a good approximation of the real terrain.

### 5.2.6 Conclusions and Future Work

With this work, we have presented a system for the procedural generation of vast landscapes with a focus on the natural and realistically-looking integration of water bodies. This is achieved by the approach of first generating river networks and lakes based on drainage basins and then the actual terrain. A quick and agile workflow is facilitated thanks to our pipeline design in which each stage is computed and

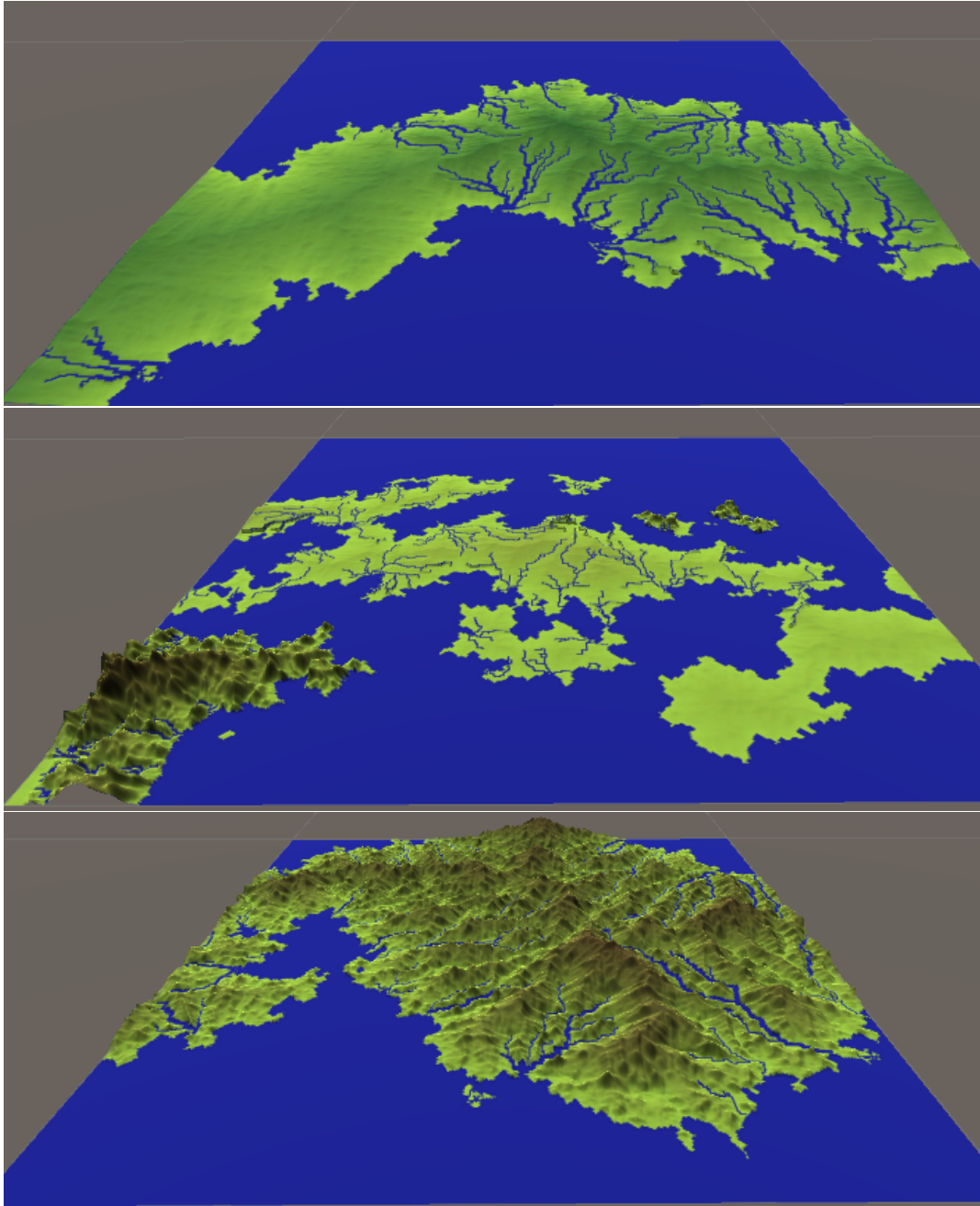


FIGURE 5.33: Example landscapes generated with our system. Note the high variability and plausibility.



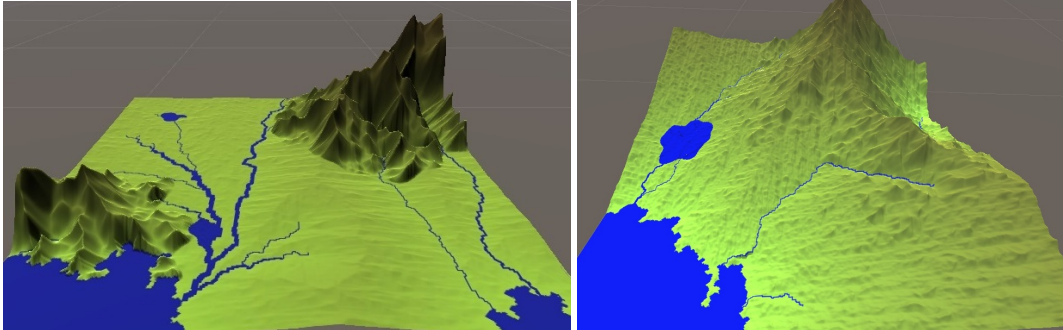


FIGURE 5.34: Examples of small-scale landscapes.

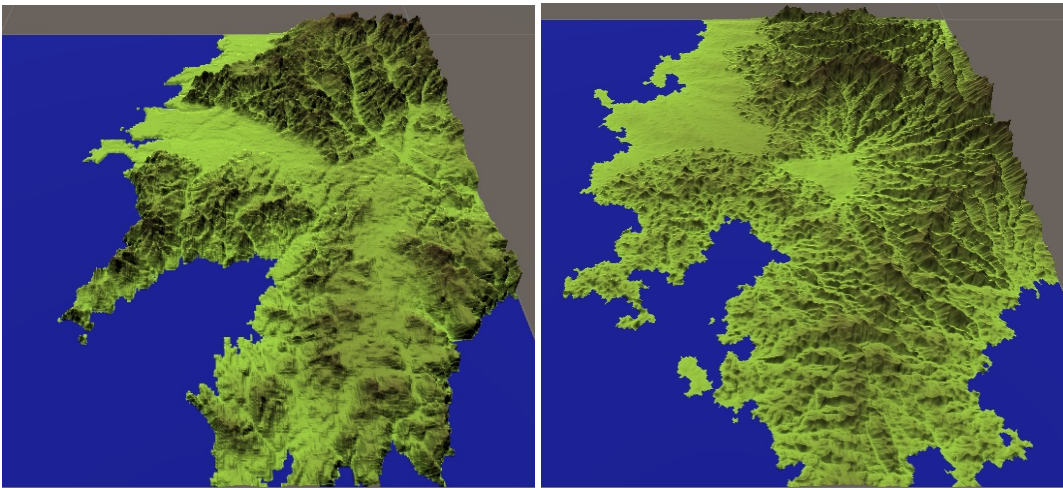


FIGURE 5.35: Comparison of a real world's terrain (left) and our recreation (right), both visualized with meshes. Note that our recreation is quite similar.

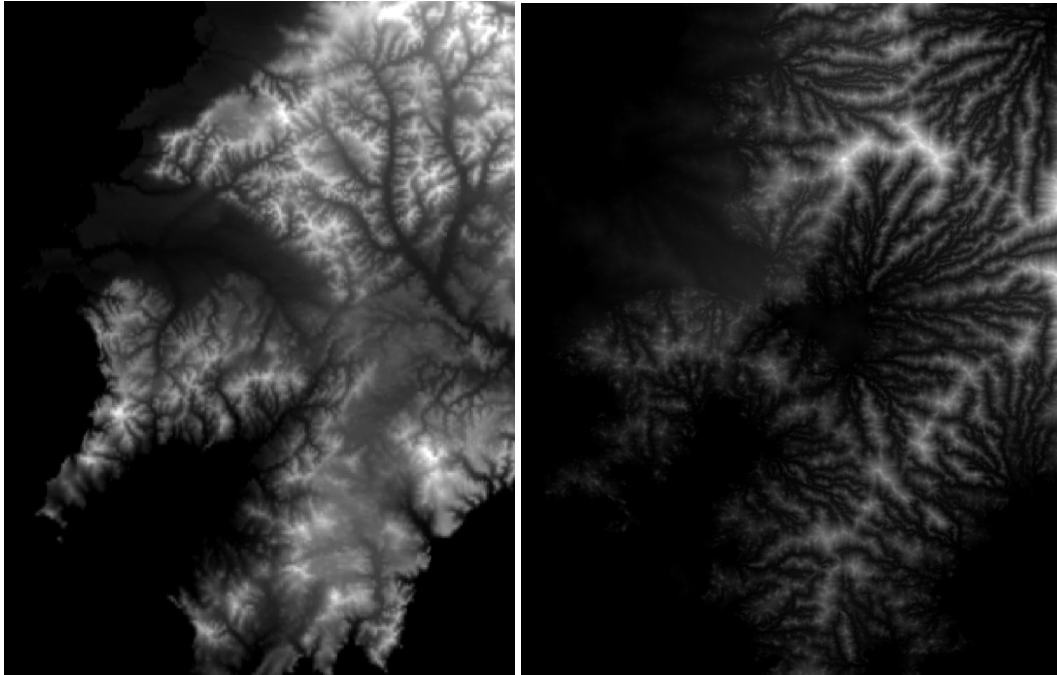


FIGURE 5.36: Comparison of a real world's heightmap (left) and our recreated terrain's heightmap (right). Our system is able to recreate the fractal nature found in the real world.

visualized in a matter of seconds. According to our performance measurements, a high-fidelity landscape (grid resolution of  $2048^2$ ) can be computed in under 25 seconds. In order to balance the amount of control, usability, and efficiency, we have designed the first pipeline stages to allow the authoring of the general landscape and its layout, while the later stages are more automation-heavy on the terrain details. Of course, our methods in the various stages of our pipeline are easily modifiable to much more or even less control, so it can be easily adapted to different needs in different workflows. Our qualitative evaluation demonstrated the great variability of our approach and a dedicated comparison with real-world terrain based on DEM data illustrated the capability to quickly create terrains strongly resembling the real ones.

In the future, one should explore the option of performing the two steps of river and terrain generation in a multi-iteration cycle that gradually refines the landscape. This would resemble the real procedures of terrain generation more closely, and thus, may produce even more realistically looking and detailed results. Another option would be to increase the landscape's variety by adding more landscape features and region types such as oxbow lakes, wetlands, and cliffs. Lastly, parallelization of some calculations could improve the computational times further.

### 5.3 Procedural Terrain Lookalikes - Generating Extraterrestrial Planetary Surfaces for VR Testbeds

In contrast to the previous two PTG systems we presented, which were mostly focused on Earth-like terrains, here, we consider a notably different scenario. Specifically, we look at procedurally generating extraterrestrial planetary surfaces based

on example DEMs – so-called lookalikes – for planning and simulation of space missions. However, this work is designed to be an outlook and show the diverse range of possible scenarios of PTG for VR-based testbeds, and thus, is only theoretical.

### 5.3.1 Introduction

Planning and conducting space missions is extremely complex, time-consuming, and costly. One example of such a mission is to autonomously explore and investigate planet surfaces such as the one on Mars. Virtual testbeds can help to accurately simulate and test many aspects of such missions in advance, thus, reducing costs and saving development time. Let us consider the mentioned example of autonomous swarm exploration. In a virtual testbed, the autonomous agents, including actuators and sensors, and their behavior can be accurately simulated, see for instance the work by Teuber/Weller et al. [374, 407]. Naturally, this requires a realistic 3D terrain model of the planet's surface. Manual modeling of detailed, large-scale terrains would be highly laborious. In case of the Mars, many DEMs, generated using satellite scans, are available. However, most are not detailed enough for a proper and meaningful simulation of unmanned ground vehicles and those that are relatively precise (HiRISE), are scarcely distributed. Procedural terrain generation could be used to solve this issue, though. Coarse DEMs could be augmented with finer details using synthetic/procedural methods such as noise or subdivision techniques, see for example the work by Li et al. [215, 214]. The drawback to this is that it is quite hard to produce realistic results with these techniques [149], especially without many detailed references, and would require a lot of laborious fine-tuning. Another possibly more promising approach could be to follow an example-based approach. Specifically, the idea would be to use one high-detailed DEM as input-example and to procedurally generate lookalikes – terrains that look similar and follow similar statistical characteristics as the input. An additional benefit with this approach would be, that many slightly varying, but still plausible and helpful, terrains could be generated to test the unmanned vehicles with, thus, increasing the robustness.

In the following, we will present and discuss two promising directions/options, including relevant related work, that follow the described example-based approach to generate terrain lookalikes.

### 5.3.2 Related Work and Promising Approaches

There exist many diverse approaches and methods for example-based PTG that could be employed for our task. In this section, we give an overview of the most relevant and promising of these methods and discuss them as well as potential issues. In fact, we identified two different general approaches into which most work can be partitioned and will consider both separately. The first one would be to employ classical heuristic-based methods from the field of image-based terrain analysis in combination with procedural/synthetic and multifractal methods, such as noise, from the field of image/texture synthesis. The second approach would be to employ generative machine or deep learning instead, e.g., GANs, and constrain them to generate similar output as the given input.

### Heuristic- and Noise-based Lookalikes

There is already a lot of research in the adjacent or more general areas of image and texture syntheses, image-based terrain analysis and classification, and example-based image generation. Hence, we propose to employ and adapt these techniques for the task of generating (martian) terrain lookalikes. This approach would consist of two steps: First, the input DEM would have to be analyzed and described, possibly even explicitly classified, using suitable and expressive heuristics and terrain descriptors. Then, the second step would entail the procedural generation or synthesis of terrains (heightmaps) that exhibit similar characteristics. This could be done using, for instance, a combination of various noise methods or other multi-fractal methods. As there would be a huge exploration space of possible parameter combinations, an efficient optimization algorithm should be used to find suitable ones.

Related works that focus on terrain analysis and classification are for example that by Zhao et al. [441], who employed object-based image analysis in order to extract terrace landforms in DEMs. For this, they used a multi-resolution segmentation method based on the slope, positive and negative terrain indices, accumulative curvature slope, coefficient of variation in elevation, terrain roughness, and slope of slope. For the subsequent classification, they used the mean and variance of the above-mentioned terrain descriptors as well as a gray-level co-occurrence matrix. The latter consisted of various variables, namely, contrast, correlation, homogeneity, entropy, and angular second moment. Stepinski and Bagaria [357] presented a stacked approach for the classification of physiographic maps in which they combine pixel-based and feature-based classification: First, the Iwahashi and Pike classifier (consisting of slope gradient, surface texture, and local convexity) is used for pixel-wise base terrain classification. Then, contextual information is considered by constructing secondary features based on the previous base classification. Concretely, the local normalized frequencies and spatial patterns (using a modified multi-scale local binary pattern operator) are taken into account for the feature generation. Eventually, clustering and segmentation are done using the “recursive hierarchical segmentation algorithm”. Another interesting work is the one by Kawale and Ferris [179], who presented a method to synthesize terrain profiles with statistical properties similar to measured example ones for vehicle durability and ride quality simulations. For this, the measured terrain is analyzed using descriptors such as the rainflow count and the international roughness index. Then, they employ an autoregressive model which maximizes the statistical conformity of the synthesized with the measured terrain. Kalbermatten et al. [171] presented a bottom-up approach for the multi-scale analysis of geomorphological and geological features in DEMs using the wavelet transform. Their approach also encompasses a filtering procedure that enhances the high-pass information from each scale and discards low-pass information. Similarly, Eisank et al. [97] employed also a multi-resolution segmentation method for DEM-based image analysis. Specifically, they used the “multi-resolution segmentation” algorithm and, for their use-case of mapping drumlins, chose the convergence index, slope height, normalized height, wetness index, and vertical distance to channel network as terrain descriptors.

Feature detection and terrain classification directly based on terrain descriptors such as slope and roughness are difficult, though, as these are bound to the local scale. In reality, features and terrain types are often nested and vary by the chosen scale. Thus, we deem it necessary to employ a multi-scale or multi-resolution approach. Which combination of the wide range of terrain descriptors works best, and

at which scale, would have to be further analyzed and empirically tested. Obviously, not all descriptors make sense in our case, e.g., the wetness index. However, for classification, we would tend to use a combination of geometrical/geomorphological features, Wavelet/Gabor transform, and co-occurrence matrix features.

For the part of procedurally generating the terrain lookalike, we can look at the field of example-based texture synthesis. There, textures are procedurally generated, often based on noise functions. Typical ones would be Perlin noise, Simplex noise, Worley noise but also Gabor noise. For instance, Galerne et al. [118] presented an approach for generating a wide range of Gaussian textures using bandwidth-quantized Gabor noise and an example input image. However, it fails to preserve larger features. Gilet et al. [129] proposed local random-phase noise, encompassing Gabor noise and noise by Fourier series, for procedural texturing with control over structural features and separate sampling in the spatial and spectral domains. Pavie et al. [274] presented locally controlled Spot noise as an extension of the local random-phase noise extending the range of representable patterns. Heitz and Neyret [142] proposed a by-example noise using a histogram-preserving blending operator that synthesizes new textures with the same appearance as an input one that is stochastic.

Again, which combination of noise functions performs best to conditionally generate the lookalike heightmaps requires further analyzation and testing, locally controlled Spot noise seems to be especially promising, but we would employ a combination of multiple noise functions over various scales and employ optimization algorithms to find suitable parameter combinations.

A principal issue with PTG, and terrain lookalike generation specifically, is the lack of established metrics to quantify the realism, or perceptual similarity of the generated images/heightmaps to the input. Usual quality/similarity metrics such as MSE, MAE, and PSNR can only be used for direct pixel-wise comparisons and do not account for perceptual, structural, or semantic similarity. The SSIM, its variants (e.g., multi-scale SSIM [399]) and some perceptual-driven metrics (e.g., perceptual image quality assessment [107], visual signal-to-noise ratio [56]) do somewhat account for the aspects of structure and perception and would be the most promising metrics for a quantitative evaluation. On the other hand, the heuristics used to describe the input terrain and its characteristics for the lookalike generation could also be employed for the evaluation, too. However, ideally, the quality of the results and similarity to the input, or, which metric/heuristic performs best in this regard, would be rated or determined by a panel of experts.

In conclusion, there exist already various methods that deal with either terrain analysis and classification, even though they are focusing on the Earth, or image/texture synthesis using noise. Thus, we find it practicable to employ, combine, and adapt these methods for the lookalike generation of Mars. Which concrete (noise) methods and heuristics work best would have to be further analyzed but the mentioned issue of scale should definitively be considered, i.e., by using a multi-resolution/scale approach. However, in the following, we also want to consider the alternative approach of using deep learning instead.

### Machine/Deep Learning-based Lookalikes

The second promising option we propose is to take advantage of machine/deep learning and generative neural networks, which showed impressive results in various computer vision and image processing tasks, including image synthesis, lately. We briefly talked about this already in the related work sections of our previous

contributions 3.3 and 5.1. One issue with this is, of course, the sparsity of available training data. As a remedy, a comprehensive and effective data augmentation workflow would be crucial. Additionally, DEMs from areas of the Earth can, and probably should, be used, too, to increase the amount of training data. Some areas on the Earth, specifically, desert-like and moderately rocky ones such as the Negev desert, some parts of Morocco, or the Canary Islands, do have similarities with (parts of) the Mars terrain [134]. In fact, such areas do get used for tests and simulations of space equipment and manned/unmanned vehicles today. Thus, an option would be to combine all training data, from the Earth and Mars, but, possibly a better, alternative would be to train on the DEMs from Earth and retrain and refine with the data from Mars.

As to which network architecture to use, we have identified three groups to consider: GANs, vision transformers, and diffusion-based models. The most widely used of these are certainly the GANs. They have been used for various image-based generative tasks, including game level and heightmap/terrain generation. For instance, Ping and Dingli [282] used conditional GANs to create tile-based game levels that follow the same design patterns as the input example. Similarly, Torrado et al. [302] employed conditional GANs and bootstrapping to more efficiently generate 2D game levels and account for longer-range dependencies. Wulff-Jensen et al. [417] trained a deep convolutional GAN on DEMs of the Alps to generate similar heightmaps. For evaluation, they used the MSE and SSIM metrics and found the GAN-based heightmaps superior to the ones generated by Perlin noise. Voulgaris et al. [390] proposed a conditional patch-based GAN to synthesize detailed terrain heightmaps only using sparse input height maps. Panagiotou and Charou [267], in contrast, employed a combination of conditional and unconditional GANs to synthesize plausible RGB satellite images and corresponding 3D point clouds. Spick et al. [351, 350] proposed using a spatial GAN in order to learn spatially-invariant features of input DEMs and generate similar terrain height maps (and corresponding RGB textures) and found it to perform better than deep convolutional GANs. Again, the SSIM and MSE were used for evaluation, as well as a subjective evaluation. Li et al. [212] used a conditional patch-based GAN focusing specifically on generating geomorphological valid features and landforms. Zhang et al. [437] also employed a conditional GAN, but in contrast to the others, trained it (using multiple discriminators) to distinguish between terrain styles, both spatially and across different scales. The generator then generates terrains by fusing multiple of these styles. Moreover, Chen et al. [59] presented an implicit periodic field network (which is based on a GAN and periodic encoding) for example-based pattern synthesis. The goal is to capture the inner statistics of the given pattern and produce tileable images recreating the patterns with smooth transitions and local variations. Lastly, we want to highlight the work by Liu et al. [224], although it does not entail a GAN. Instead, they proposed a perception-driven example-based approach to procedural texture generation using a new principle component analysis-based CNN. The idea is that the network extracts the textures features, predicts the perceptual qualities of the input (empirically collected and labeled, e.g., contrast, uniformity), and a procedural/noise model including its parameters that is best suited to recreate similar output.

An interesting idea is to combine deep neural networks such as GANs with evolutionary algorithms. For instance, Liapis et al. [219] proposed to alternative between the exploration of novel and diverse content using constrained novelty search and transforming the found content, and adapting the fitness function using deep denoising autoencoders. Thus, the results get iteratively refined. Other

works such as the ones by Volz et al. or Irfan et al. [389, 155] employ a combination of GANs/variable autoencoders and latent variable evolution to search for novel content with desirable attributes. This approach seems to be particularly promising as it should make the generation process more efficient and targeted.

The second group of generative networks is diffusion-based networks, which recently outclassed GANs in image synthesis tasks [84, 146]. In contrast, they are likelihood-based and do not suffer from issues such as mode collapse and training instabilities. They are significantly slower and extremely memory-intensive, though. To make them more efficient and enable higher-resolution synthesis, Rombach et al. [306] proposed applying the diffusion model to the latent space of pre-trained U-Nets. However, to our knowledge, they were not applied to DEMs/heightmaps yet. Hence, it would be highly interesting to evaluate their performance regarding DEM-based terrain (lookalike) generation.

The third group we want to consider is vision transformers. Originally designed for natural language processing tasks, they rapidly grow in popularity and show also promising results in computer vision tasks, including, in some cases, generative ones. For instance, Chang et al. [57] proposed a bidirectional transformer for image synthesis that shows impressive results on image generation and manipulation tasks. Park and Kim [270], in contrast, presented a transformer-based generator for image synthesis that uses style vectors. They propose to combine it with either the new self-attention mechanism “Linformer” by Wang et al. [397] or StyleGAN2 [177] for more efficient high-resolution image generation. As with the diffusion-based models, we are not aware of any works applying vision transformers on heightmaps/DEMs for example-based PTG. Thus, we would like to investigate how they, solely or as an encoder in a GAN, perform and if they can beat classic convolution-based GANs.

Recently, several works such as the one by Khalifa et al. [184] also proposed the idea of using reinforcement learning by interpreting the procedural generation task as a Markov decision process and iteratively selecting actions that maximize the expected result’s quality. Applied to PTG, the given terrain could iteratively be transformed towards a specified goal. The interesting questions are what suitable transformation actions would be and how to specify the goal (a terrain lookalike). Regarding the former, we do not think pixel-wise height adjustments would make sense but would envision either localized modifiers (similar to the ones used in 3D modeling and sculpting tools) or pre-defined global noise functions or filters. One issue with this approach could be the vast action space, though. For the latter question, the goal specification, we have the same challenge as with all other (deep learning) methods, namely, how to objectively evaluate the resulting lookalikes. In deep learning-based image generation, the Fréchet inception distance and its variants, such as the conditional Fréchet inception distance, are commonly used to assess the (class-conditional) quality (realism, diversity) of the resulting images. As with the first presented approach of using heuristics, a panel of experts would be ideal to either directly assess the results or determine the most suitable metric.

Our recommendation for the deep learning approach would be to first investigate and compare the performance of one state-of-the-art model of each group for this specific task of DEM lookalike generation before going deeper and optimizing and tuning the model further.

All in all, both of the presented and discussed options for the example-based Mars-terrain lookalike generation seem to be viable and promising but need further investigation and prototyping. Objective evaluation of the quality and similarity of the result is a shared issue that needs to be dealt with, e.g., by a panel of experts.

We suspect that the heuristics and noise-based approach might be more targeted and better to control but require more tuning and is worse at generalization than the deep learning approach. Possibly, both approaches could be combined for the best performance, e.g., by taking the geometric and geomorphological heuristics into account when using a generative neural network.



## Chapter 6

# Applications for Multi-User VR in the Medical Field

Throughout this thesis, we discussed various aspects of multi-user VR and telepresence applications, e.g., generating detailed virtual environments, live-capturing and streaming of remote scenes and avatars as well as enhancement, reconstruction, and rendering of these, too. In this chapter, we want to present two applications of multi-user VR for the medical field. This area is especially interesting, as, there exist many scenarios and applications that can profit from multi-user VR. The first one is the volumetric depiction of and interaction with medical data such as CT scans. Although there exist specialized programs for this task, they usually do not support immersive exploration through VR or are limited to single-user usage. In Section 6.1, we, therefore, present an easy-to-use and expandable system for volumetric medical image visualization with support for multi-user VR interactions. The main idea is to combine a state-of-the-art open-source game engine, the Unreal Engine 4, with a new volume renderer.

Another application domain in the medical area in which VR and the 3D representation/ visualization of medical data are employed to great benefit is VR anatomy atlases for anatomy education. These provide students with detailed, interactive 3D human anatomical models that can be repeatedly explored in a virtual environment. Often, they get also annotated with additional information. As they usually only support individual learning and there is sparse research on the effectiveness of collaborative anatomy learning in VR, we developed a collaborative VR anatomy atlas and conducted a user study to compare the learning progress and usability between individual and collaborative use. More information about this can be found in Section 6.2.

## 6.1 Volumetric CT Data Visualization for Collaborative VR Environments

In clinical practice, medical imaging technologies such as CT or MRI have become an important and routinely used technique for diagnosis. Advanced 3D visualization techniques of this data, e.g., by using volume rendering, provide doctors with a better spatial understanding for reviewing complex anatomy. However, programs for the visualization of medical imaging data, are usually limited to exactly this topic and can be hardly extended to new functionality, for instance, multi-user support. In contrast, immersive VR interfaces like tracked HMDs and natural user interfaces could provide doctors an easier, more immersive access to information and support collaborative discussions with remote colleagues.

Thus, in this section, we propose a multi-user VR system based on the state-of-the-art game engine that includes a custom volume rendering solution to directly visualize volumetric medical data such as CT images. The underlying game engine basis guarantees the extensibility and allows for easy adaption of our system to new hardware and software developments. In our example application, remote users can meet in a shared virtual environment and view, manipulate and discuss the volume-rendered data in real time. Our new volume renderer for the Unreal Engine 4 is capable of real-time performance, as well as, high-quality visualization.

The work presented in this section is based on our published paper PC3 in Appendix A.

### 6.1.1 Introduction

Computed tomography is an x-ray-based medical imaging procedure that produces sequences of 2D cross-sectional images (called slices) of solid objects like the human body and therefore allows the visualization and inspection of the inner parts. It is a vital examination tool in medicine, especially for radiologists, and is widely used in clinical practice. Its use cases range from diagnosis and therapeutics to preventive medicine and screening of diseases. CT images are, for example, commonly used for visualization purposes in tumor board reviews or for postmortem imaging in forensic pathology. 3D visualization of the CT data is rarely taken advantage of yet. However, it is slowly getting more important. Due to rising processing power and continuous research in algorithms and rendering techniques, faster and more advanced 3D visualization techniques are developed. The main benefit is the more intuitive, three-dimensional visualization of the data. This makes it easier and faster to get an overview of the data and an understanding of the spatial relations, volumes, and general layout of the depicted objects. This is helpful for analyzing complex anatomy or conveying medical situations in an easy-to-understand way. Typical 3D visualization techniques are maximum/minimal intensity projection, surface shaded display, also called indirect volume rendering, and direct volume rendering (DVR). We briefly touched up on this topic in Section 2.2.4, but to elaborate, surface shaded display shows opaque three-dimensional surfaces, called isosurfaces, of specific objects or organs in the volume data determined by a density-dependent segmentation. To render the isosurface, a polygonal model has to be constructed first. Its main advantage is the high performance, however, the binary classification may lead to incorrect classifications and artifacts [110]. DVR does not face these problems, as it is not limited to this binary classification. Instead, DVR accounts for the possibility of multiple tissue types per voxel and maps the densities to opacities and colors using transfer functions. This results in a semi-transparent rendering [77, 96].

In Section 1.1, we already mentioned that currently, both 2D and 3D CT reconstructions are typically viewed on 2D screens or projectors, which limits the advantages of volumetric visualizations. On the other hand, VR devices such as the HTC Vive become popular in many fields as they provide immersive stereoscopic visualizations with intuitive user interfaces and novel cooperative multi-user capabilities. VR offers a natural progression over previous 2D telepresence tools and leads to a new quality of collaborative work, as users can meet and intuitively interact with virtual objects as well as with each other in a shared virtual 3D environment. This makes VR an important tool for the entertainment industry but also for industrial, educational, and medical applications. For example, a current trend is to use VR for simulators in which users can be trained and educated realistically and in a safe

virtual environment (e.g., laparoscopy, heart surgery, and even orthopedic operations [173]). These benefits and the increasing display resolutions of newer headsets make VR in general, and multi-user VR particularly, well suited for use cases like inspection and discussion of volumetric medical data and corresponding 3D visualizations as part of diagnosis or pre-operative planning [257].

VR applications and their virtual environments are typically created and powered by 3D graphics engines like Unity or the Unreal Engine which provide features such as high-quality graphics and automatic VR integration. However, they are usually mesh/polygon-based and, out of the box, do not support volume rendering.

We propose a system based on the Unreal Engine 4 in which multiple users can collaboratively inspect and interact with volume-rendered CT data in real-time within a VR environment resembling an operating room. For this purpose, we combine mesh- and volume rendering into an immersive multi-user application. This includes a custom direct volume renderer for the Unreal Engine and several optimization and lighting techniques to achieve real-time performance as well as a good visualization quality. Additionally, we have developed a custom pipeline for processing CT images allowing easy and effective visualization of multiple windows in parallel.

### 6.1.2 Related Work

Volume rendering is a promising tool for medical visualization as it proved to be useful for planning of surgical treatment of nasal bone fractures [348], acetabular fractures [395], virtual endoscopy [190] or the visualization of complex anatomy such as the ossicular chain in chronic suppurative otitis media [138] or for visualizing the relationship between stent-grafts and arterial vessels [365]. Recently, a DVR approach for serial PET–CT scans that preserves anatomical consistency was presented by Jung et al. [170]. Key elements are an automated serial transfer function optimization, an interactive serial segmentation algorithm and a GPU implementation. The high computational effort of DVR can be mitigated by algorithmic optimizations, e.g., early ray termination and empty space skipping [313]. Also, the visual quality can be improved, e.g., by applying local ambient occlusion [145]. Berger et al. [22] have shown that the novel, more complex cinematic rendering technique (based on Monte Carlo path tracing) provides a superior visualization to the classic volume rendering using ray casting, however, the significantly slower computation is still a challenge. Ryan Brucks [39] developed a custom volume rendering implementation for the Unreal Engine 4, however, it is only rudimentary and not designed for medical data, leading to artifacts.

Several evaluations show that VR can be beneficial in a wide range of medical applications, foremost simulators for training different surgical procedures [200, 289]. For example, Kozak et al. [189] introduced a virtual reality retina surgery simulator using optical coherence tomography data. Additionally, a study by Feudner et al. [109] concluded that VR-trained students achieved a significantly higher wet-lab performance of capsulorhexis. Often, medical imaging plays a central role in these applications: e.g., Maloca et al. [233] proposed an OpenGL-based immersive VR system for real-time volume rendering of optical coherence tomography data that renders the volume data using point clouds in a virtual environment. An accompanying study suggested that it could be helpful for education and preoperative planning. Similarly, Scholl et al. [321] developed a medical VR application for 3D visualization based on volume rendering that allows to interact with the volume using, for instance, a dynamic clipping plane and modifiable transfer functions.

Real-time performance is achieved by the use of several acceleration and optimization techniques, including lens matched shading, shadow ray diffuse culling, and semi-adaptive sampling. Adams et al. [1] used the Unity 3D engine to develop an immersive VR application for medical imaging in which CT images and corresponding, segmented 3D models can be viewed and manipulated. Magdics et al. [229] also used Unity to develop an educational VR application in which DVR, including volumetric ambient occlusion, is used for visualizing Nasal Cavities based on MRI data. Faludi et al. [106] presented a VR application that uses not only DVR but also haptic rendering of medical data. However, none of these systems support multiple users or collaborative work, which is another popular and promising research area.

Regarding collaborative medical VR, Cecil et al. [52] developed a system for orthopedic surgery, specifically, less invasive stabilization system surgery. It also includes a haptic interface. Similarly, Paiva et al. [265] presented a VR simulator for surgical team training, in which users can assume various roles in a virtual operating room (e.g., surgeon, anesthetist, etc.) or join as an observer. The users get visualized by full-body avatars and Leap Motion-based hand-tracking is employed for detailed interactions. Chheang et al. [63] proposed a promising collaborative VR system for planning and simulation of laparoscopic liver surgery. Interactive, virtual 3D organ models are reconstructed based on patient data and visualized in a virtual operating room. In addition to the VR controllers, surgery joysticks can be used for training. Christensen et al. [66] positively evaluated the feasibility of team training in VR for robot-assisted minimally invasive surgery, and Elvezio et al. [99] designed a VR system for collaborative symmetric and asymmetric interactions and found that low latencies (below 15 ms) are crucial for effective collaboration. These works, however, do not feature 3D visualization of CT data.

### 6.1.3 Proposed Approach

The goal of our system is to combine the benefits of collaborative VR and medical 3D visualization into an immersive, interactive application based on a modern, extensible open-source 3D game engine, specifically the Unreal Engine 4. As the engine does not support (direct) volume rendering out of the box, we have developed and integrated a ray-marching-based volume renderer, based on Ryan Brucks' rudimentary implementation [39], focusing on a good trade-off between speed and visual quality.

We have decided to use the Unreal Engine 4 for several reasons: first, it is known for its high graphical fidelity, second, it supports most available VR devices like the HTC Vive with a platform-independent interface, and it has networking capabilities included. Moreover, due to its open-source implementation, it can be easily extended with native C++ programming but also offers an easy graphical programming interface via Blueprints. We decided to directly benefit from Unreal's networking architecture, hence, we use a client-server model enabling users to host and join sessions via a lobby system, whereby the first client acts also as a server. An overview of the whole system design is shown in Figure 6.1.

An overview depicting all the necessary steps to volumetrically render the CT data in the UE4 is shown in Fig. 6.2. First, the CT data, which usually uses the DICOM file format, has to be read. Then, the CT data requires a preprocessing step before it can be effectively used. Moreover, we combine the data into 2D sequence maps before eventually importing them into the Unreal Engine. The next step is to construct an octree for optimization purposes. The processed data can then be

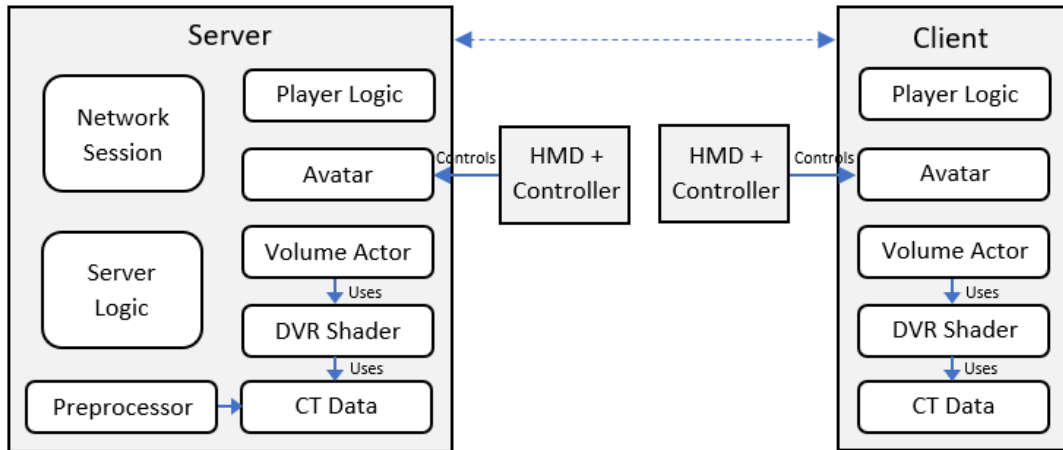


FIGURE 6.1: System architecture of our application. The first client acts also as a server.

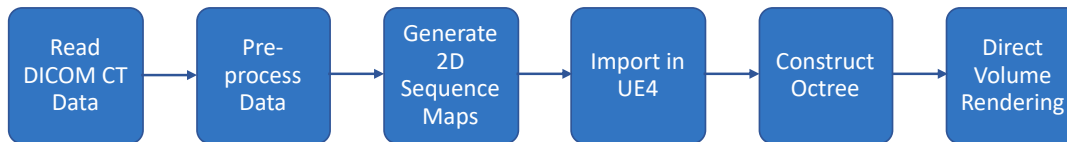


FIGURE 6.2: Pipeline of our UE4-based direct volume rendering approach of CT data.

rendered seamlessly into the polygonal scene using our shader-based DVR solution. We have integrated several lighting techniques such as local ambient occlusion to improve the visual quality. Our DVR approach achieves real-time performance guaranteeing a smooth VR experience. In the following, we will describe the individual parts of our system in detail.

### Direct Volume Rendering

In order to visualize the CT data in our Unreal Engine-based virtual environment, we opted for a DVR approach based on ray marching. Our pipeline is specifically designed for the visualization of CT data, thus, the first step is to read and process the CT DICOM files (which is the usual format) in a preprocessing phase. For this purpose, we wrote scripts utilizing `pydicom` to parse the relevant information, see Table 6.1, and compute the density values (in Hounsfield units (HU)).

To map the density to opacity, we employ multiple, freely adjustable, default windows with corresponding transfer functions. The advantage of having multiple windows is that each feature captured by a window can be visualized with high

TABLE 6.1: Relevant DICOM attributes parsed from the CT data.

Keyword	Ex. Value
Slice Thickness	3.0
Rows	512
Columns	512
Rescale Intercept	-1024
Rescale Slope	1
Window Center	[40, 700]
Window Width	[350, 1800]

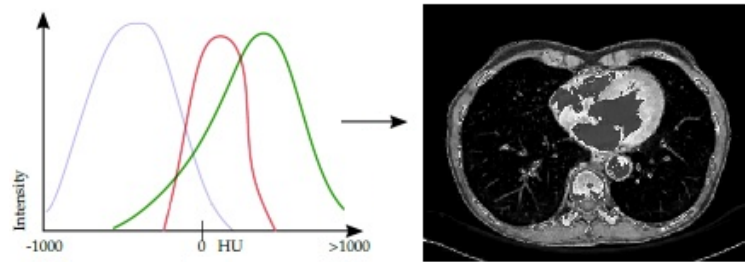


FIGURE 6.3: Window blending according to the RADIO algorithm. Left: bone, lung, soft tissue windows. Right: blended CT image.

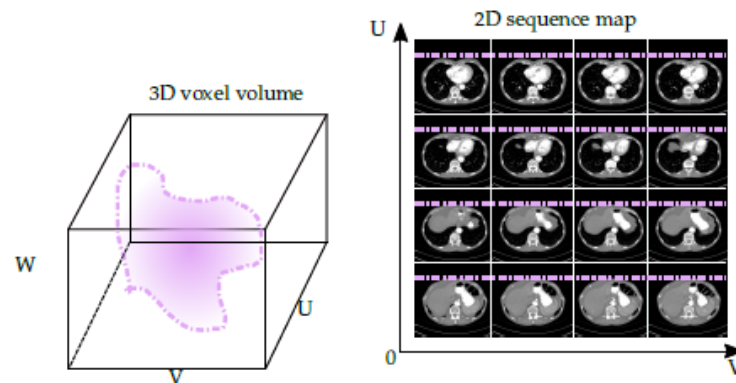


FIGURE 6.4: Right: sequence map of CT slices. Left: corresponding reconstruction in the shader.

contrast. To store the windows in a single grayscale image (8 bit) we decided to blend the windows similar to the RADIO algorithm by Mandell et al. [234], which maintains the relative attenuation relationships between the fundamental anatomic densities and thus accommodates radiologists and their expectations. Figure 6.3 depicts the underlying concept. However, any other blending algorithm would be compatible too.

Additionally, the volumetric data set has to be transformed into a format suitable for import and further processing in the Unreal Engine, therefore, we arrange the individual 2D slices of the volume sequentially into sequence maps, sometimes also called a flipbook (see Fig. 6.4). Each CT slice is then a sub-image in the sequence map. This is a typical approach to store volumetric data due to common limitations such as 3D engines not supporting volume textures, and pixel shaders heavily relying on textures as an input source. An overview of all steps done during the preprocessing phase is shown by Algorithm 10.

We have implemented the ray casting directly in a pixel shader. We use a unit cube as a geometrical proxy mesh and reconstruct the volume coordinates from the generated sequence maps. In order to avoid artifacts of box-aligned samples (left diagram in Fig. 6.5), we align the first sampling points to stacked view-aligned planes instead (middle diagram in Figure 6.5). Additionally, we precompute the sampling step length and the maximal number of samples fitting in the volume outside of the ray casting loop to reduce overhead. The calculation is based on the CT data set's proportions, the ray's accordingly adjusted starting position, and a user-adjustable factor allowing for arbitrary changes to the sampling rate. More details about the aforementioned handling of the sequence maps and the general sampling procedure

**Algorithm 10** Preprocessing stage

---

**Require:** DICOM files, WindowSettings[]  
*BlendedWindows*[*Slices.Num*], *GradientMaps*[*Slices.Num*]  
*Slices*  $\leftarrow$  *ReadDicom*(*DICOMFiles*)  
*Slices*  $\leftarrow$  *Sort*(*Slices*)  
**for** each *Slice* in *Slices* **do**  
  *Tags*  $\leftarrow$  *ReadTags*(*Slice*)  
  *Densities*  $\leftarrow$  *CompDensity*(*Tags*, *Slice*)  $\triangleright$  in Hounsfield Units  
  *windows*[]  $\leftarrow$  *CompWindows*(*Densities*, *WindowSettings*[])  $\triangleright$  3 windows  
  *BlendedWindows*[*Slice*]  $\leftarrow$  *BlendWindows*(*windows*[])  $\triangleright$  mod. RADIO alg.  
  *GradientMaps*[*Slice*]  $\leftarrow$  *CentralDifferences*(*BlendedWindows*[*Slice*])  
*SeqMap<sub>d</sub>*  $\leftarrow$  *CompSeqMap*(*BlendedWindows*)  
*SeqMap<sub>g</sub>*  $\leftarrow$  *CompSeqMap*(*GradientMaps*)  
*Octree*  $\leftarrow$  *ConstructOctree*(*SeqMap<sub>d</sub>*)  $\triangleright$  BONO octree  
*OctreeMap*  $\leftarrow$  *EncodeOctree*(*Octree*)

---

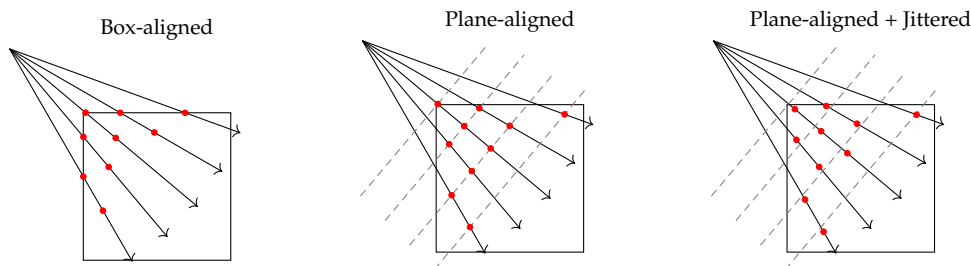


FIGURE 6.5: Sampling positions in the volume. Left: in the naive approach the start sample positions align with the box mesh and cause patterned artifacts throughout the volume. Middle: sampling positions on equidistant view-aligned planes. Right: the sample positions are additionally jittered along the ray axis.

can be read in Ryan Brucks' volume rendering guide for the Unreal Engine 4 [39]. We provide the possibility to apply stochastic jittering and a  $2 \times 2$  ordered grid super sampling anti-aliasing (SSAA) to improve the visual quality (see the right diagram in Fig. 6.5). Moreover, we achieve trilinear interpolation using a combination of bi-linear interpolation of the textures and linear interpolation between slices.

Regarding convincing yet fast shading and shadowing, we opted to implement a couple of different and not-too-complex local illumination methods and compare the visual results. Firstly, at each sample position, we cast shadow rays to determine the amount of occlusion. For this purpose, we dynamically track the position of multiple light sources. This method enables proper self-shadowing from multiple dynamic lights, however, is rather computationally expensive. Therefore, we lower the shadow rays' sampling frequency in contrast to the primary rays'. Secondly, we implemented the classic Blinn-Phong shading model that is evaluated at each sampling position. It is rather cheap to compute and enables local lighting approximation by diffuse and specular reflections which can be configured on a per-material basis. We approximate the needed surface normals, which are not present in CT data, based on the local gradient using the central differences technique in the preprocessing phase. Lastly, we also implemented volumetric local ambient occlusion (LAO). Here, the sampling point is shaded based on the amount of occlusion, which is estimated by the opacities of the local neighborhood. This method can be used to

prevent full shadows, which may obscure fine details. Another advantage is that it is not based on gradients, which are often not well-defined (e.g., in homogeneous regions) and susceptible to noise. Algorithm 11 outlines the raycasting process.

---

**Algorithm 11** Shader-based raycasting (without octree)
 

---

**Require:**  $SeqMap_d$ ,  $SeqMap_g$ , proxy volume  $Proxy$

**for each**  $Pixel$  (in parallel) **do**

$Start \leftarrow CompStartPos(Camera, Proxy)$

$Start \leftarrow AlignAndJitter(Start)$

$Step \leftarrow CompStepLength(SliceDepth, NumSlices, UserFactor)$

$MaxSamples \leftarrow CompMaxSamples(Proxy, Start, Step)$

$CurPos \leftarrow Start; SampleID \leftarrow 0$

**while**  $SampleID < MaxSamples$  and  $AccumOpacity < 1$  **do**

$Opacity \leftarrow SampleTriLin(CurPos, SeqMap_d, Step)$

$AccumOpacity \leftarrow AccumOpacity + Opacity$

$Color \leftarrow CompSampleColor(Opacity)$

**if**  $ShadowRays$  set to  $True$  **then**

$Occlusion \leftarrow ShootShadowRays(CurPos, Lights[])$

$Color \leftarrow Shade(Color, Occlusion)$

**else if**  $LAO$  set to  $True$  **then**

$Color \leftarrow LAO(Color, CurPos, SeqMap_d)$

**else**

$Color \leftarrow BlinnPhong(Color, CurPos, Camera, Lights[], SeqMap_g)$

$AccumColor \leftarrow CompColor(Color, AccumColor, AccumOpacity)$

$CurPos \leftarrow CurPos + Step$

---

To increase the performance, we reduce the number of samples being taken by early ray termination and empty space skipping using an octree. Early ray termination means that if the front-to-back accumulated opacity reaches 100 percent, the rest of the volume further back is obscured and does not need to be sampled any more, thus, the loop is terminated “early”. Moreover, many regions in the CT data are possibly empty, in this case, for instance, being just air. Therefore, they don’t need to be sampled repeatedly. The octree partitions the space hierarchically into homogeneous regions based on the density. Regions with a near-zero density are practically empty and can be skipped in the sampling process, see Figure 6.6. We construct the octree using a pointer-free branch-on-need strategy and encode it in a texture during the preprocessing phase, as the data is static. During sampling, the octree is traversed top-down similar to the parametric approach described in [297].

### Collaborative VR

Generally, we made use of the Unreal Engine’s polygonal and stereo rendering capabilities and VR support to build our application. To create a believable virtual environment, thus, enhancing the immersion and the experience for the users, we build a 3D scene resembling an operation room in which the users can interact. Similarly, users are represented by static mesh avatars modeled after doctors in medical outfits. Our avatars consist of separate models for the head and hands; their corresponding positions are tracked directly by the HMD and the accompanying controllers. To avoid issues with possibly faulty and distracting animations by inverse kinematics, we refrain from using full-body avatars. Each user can be identified by



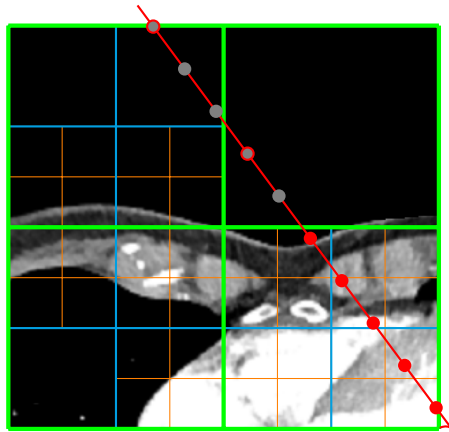


FIGURE 6.6: Empty space skipping using an octree (visualized in 2D). The CT data is hierarchically segmented (rectangles) and gets sampled by a viewing ray (red arrow). If an empty space cell is sampled (gray node with red outline), all further samples in the same cell get skipped (gray nodes).

a personal name shown over the avatar. Figure 6.7 shows a session with three users inspecting the CT data in the virtual operating room.

We included a lobby system with which users can create or search for active sessions, or alternatively join one via a known IP, thus, enabling multiple of these virtual shared environments to exist in parallel. The left image in Fig. 6.8 shows an example of the lobby screen. Also, although VR usage is our main focus, VR and non-VR users can mix and collaborate without restriction as we have implemented movement and interaction metaphors for both of them. For example, we implemented physical 3D buttons placed in the scene for VR users and keyboard shortcuts for non-VR users to manipulate the properties of the 3D visualization, see the right image in Fig. 6.8. To reduce the latency between user input and perceived action, which has been shown to be crucial for a positive user experience in previous studies [393], all (inter)actions from users are executed locally first, before being sent and replicated on the server, from which they are finally broadcasted to all remaining users. Furthermore, the Unreal Engine provides some additional latency optimization techniques which help to minimize and stabilize the time needed for communication between client and server.

As a locomotion metaphor for VR users, we decided to use the classical teleportation approach, in combination with room-scale locomotion, as it minimizes the occurrence of cybersickness [67]. The teleportation works in the classical fashion, however, users can choose the new orientation via the controller's trackpad, and teleporting on top of objects or out of the room is being prevented. A problem arising from using teleportation in a multi-user environment is that the actual process of vanishing and reemerging somewhere else will be confusing for observers as it resembles the typical effects of a slow network connection or network errors. Therefore, we have implemented a particle effect, to highlight the deliberate action of the teleportation process.

To allow for collaborative work between users, we replicate not only their avatars but also the complete state of the 3D visualization of the CT data, making it a single shared object in the scene that is rendered from the individual users' viewpoint. It can be grabbed, moved, and rotated freely and naturally using the controller for optimal (re)view (see Fig. 6.9). Non-VR users, however, can rotate the object via an

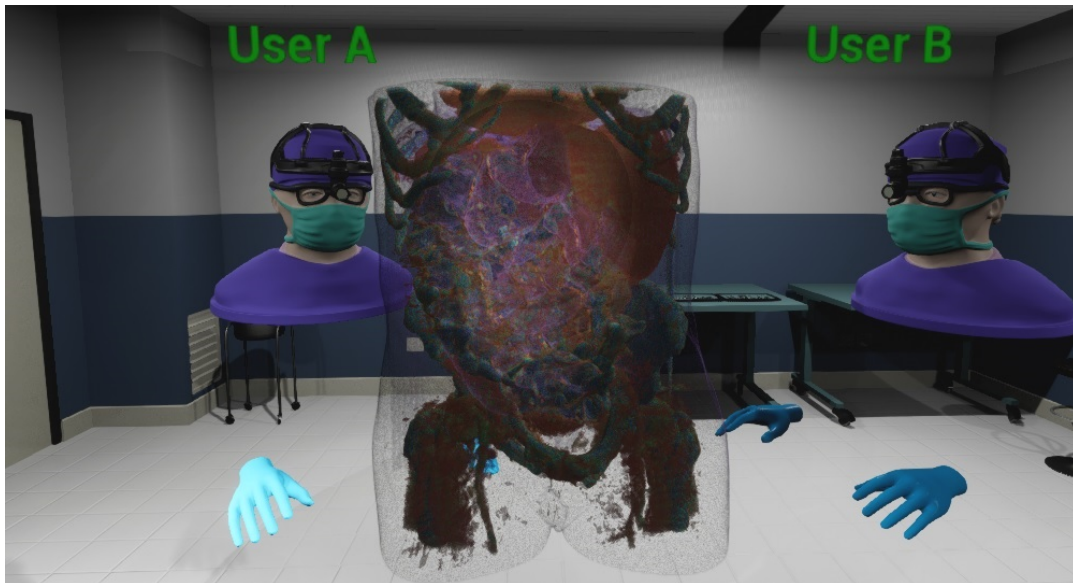


FIGURE 6.7: Several networked users inspecting the 3D visualized CT data in a shared virtual environment.

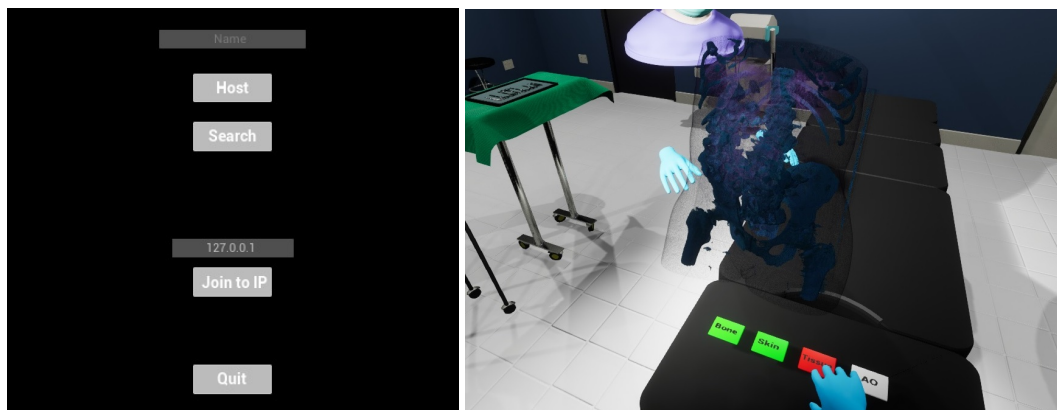


FIGURE 6.8: Left: A lobby system provides access to hosting or joining multi-user sessions. Right: Physical buttons enable the real-time switching of the 3D visualization's properties, e.g., the lighting mode or transfer function.

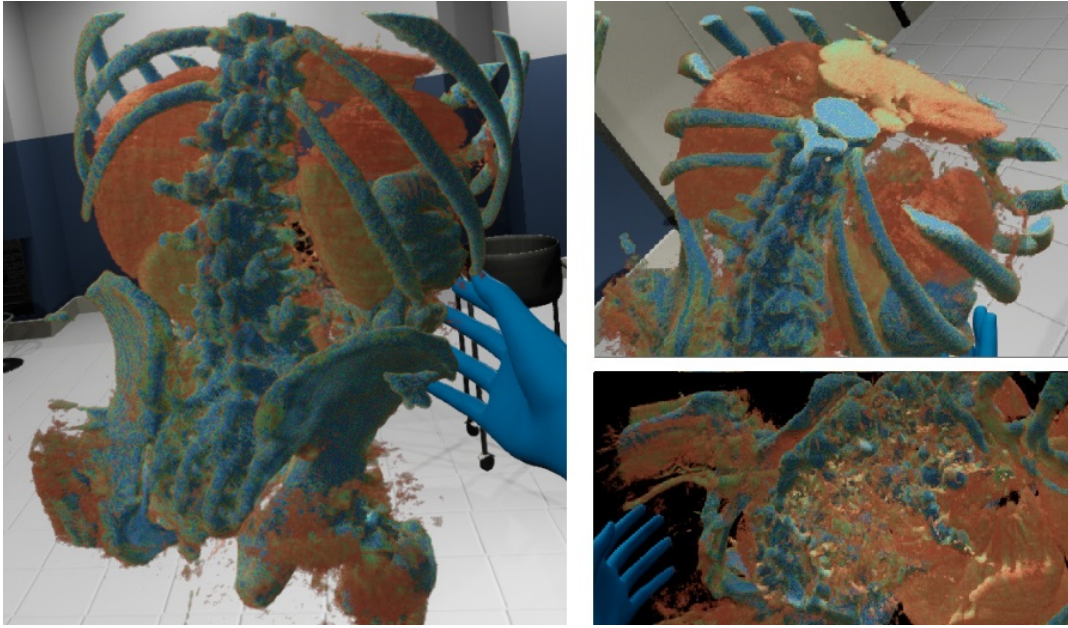


FIGURE 6.9: Several images illustrating how the 3D visualization can be grabbed and freely moved and rotated for a better view. The background was hidden in the image in the bottom right.

orbiting mode. To keep it simple, we do not restrict concurrent manipulation, which internally would be executed sequentially, as users can coordinate themselves. The replicated hands in VR make it easy to point to specific spots or areas in the 3D visualization and to make gestures, which help in discussing the data, showing findings, or planning interventions.

In addition to the 3D visualization of the medical data, users in our application have the possibility to view accompanying 2D images, e.g., the raw CT data, on a virtual TV screen in the OR. This may be useful if there is a need to quickly check for specific fine details not visible in the 3D visualization. Finally, the complete scene but the 3D visualization can be dynamically hidden, resulting in a black background, for an undistracted contrast-rich view.

#### 6.1.4 Results

We have evaluated the quality as well as the performance of the main aspects of our approach. To show the quality of our volumetric renderer, we visually compare our results to two competing visualization tools. Additionally, we did extensive measurements regarding the performance under various conditions, e.g., different lighting models and optimization methods. For the evaluations, we used several real-world CT data sets obtained by a hospital. The number of slices varies between the data sets and ranges from 47 to 317. We have developed and tested our work based on the Unreal Engine 4.22. Figure 6.10 shows our volume renderer with different active windows. In the left image, only the bone window is applied. The middle image depicts, among others, inner structures of the liver, small intestine, colon, and skin. Finally, in the right image, all three windows (the third one being soft tissue) are simultaneously visualized. Our DVR is able to effectively render single materials like bone as well as compositions of multiple materials simultaneously, and thus, the complete range of CT data. This helps in conveying the spatial relationships between organs and getting a good understanding of the data.

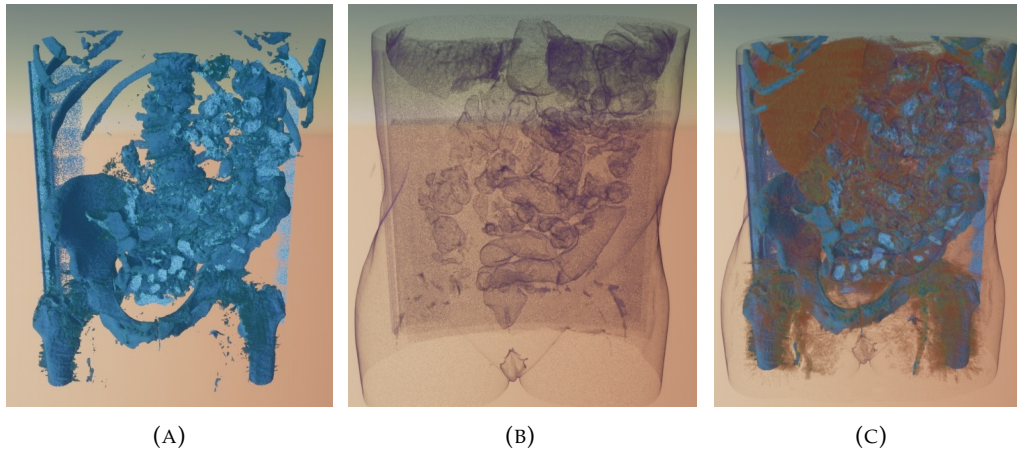


FIGURE 6.10: Our volume renderer applied to a CT data set using different windows: bone (a), bowel and skin (b), and soft tissue, combined with the previous windows (c).

Figure 6.11 depicts our renderer with the different lighting settings. The left image shows a bone window using only shadow rays. Self-shadowing can be seen which helps in conveying depth, however, because of the limited sampling rate for shadow rays, the shadows are coarse and imprecise. In the middle image, we switched on the Blinn-Phong lighting model. A possible issue with this technique is, that, depending on the position of the light source relative to the visualized object, areas may lie completely in the shadows, and thus, can be hard to inspect if no additional ambient lighting is applied. The right image, however, shows the combination with the LAO technique. This combination circumvents the problem of full shadows and results in the best lighting. The transition between being in complete light and full shadow is the most fine-granular and accounts for the local neighborhood providing the best depth perception and understanding of object shapes.

Figure 6.12 illustrates a comparison of our renderer (first image) with the common visualization tools RadiAnt DICOM Viewer in the standard 3D volume rendering mode (second image), and the Visualization Toolkit (VTK) with maximum intensity projection as a composition scheme (third image). The comparison shows that our renderer generates visualizations that are very effective in conveying a perception of depth and giving a clear and understandable overview of the data set as a whole. At the same time, our renderer produces precise, plastic visualizations of the individual materials. VTK uses maximum intensity projection which results in relatively flat images with missing details. The advanced lighting and shading of our renderer make the assessment of the spatial relations between the objects easy. Although the results by RadiAnt are very good too, they tend to exhibit slightly stronger artifacts and a simpler shading is used.

Fig. 6.13 depicts another comparison of our volume renderer with RadiAnt and VTK, similar to the one in Fig. 6.12, but with a different dataset. The results are similar: our rendering solution produces high-quality results that provide a good depth perception.

In Figure 6.14, we show a comparison of multiple optimization techniques that we have implemented to reduce artifacts. The first image (A) shows the basic DVR without any optimization, which results in strong artifacts. Using pre-integration (image (B)), the artifacts can be reduced significantly and the individual organs can be seen more clearly. The remaining artifacts can be reduced further using either

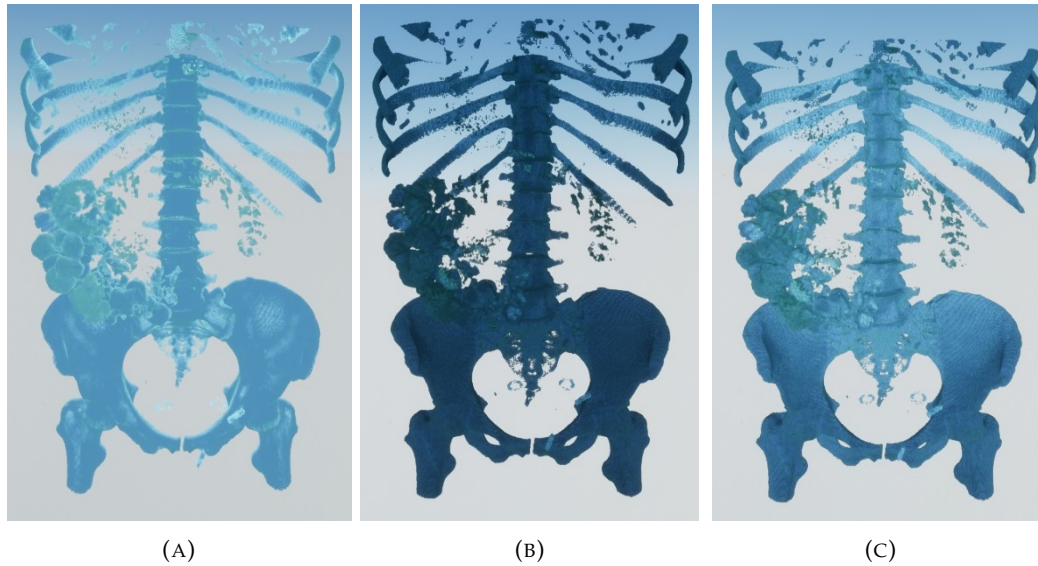


FIGURE 6.11: Our volume renderer using different illumination methods: shadow rays (a), additional Blinn-Phong lighting (b), both combined with LAO (c). As can be seen, the latter enhances depth perception through superior shadowing.

stochastic jittering (C), or SSAA (D).

A performance evaluation was done on a PC with Windows 10, Intel Core i7 4790 CPU, Nvidia Titan V graphics card, 32GB of system memory, and a Full HD monitor. To perform the measurements, we used the native GPU profiler of the Unreal Engine and took the average of multiple runs.

The theoretical computational complexity of our DVR is  $\mathcal{O}(m \cdot n \cdot l)$  with  $m$  being the number of primary rays,  $n$  being the number of samples taken per ray, and  $l$  being the number of light sources. The number of rays is directly dependant on the screen resolution (and SSAA), therefore, theoretically exhibiting a linear growth with the number of pixels. Though, by the inherently parallel computation by the GPU, the complexity is lowered. The number of samples per ray can be chosen accordingly to the the number of slices. In practice, however, many factors have a significant impact on the performance, e.g., the chosen lighting method, the data itself, as well as its distribution in the volume.

Figure 6.15 shows the performance of our renderer and the influence of factors such as the number of slices and different lighting methods. As expected, having more slices leads, generally, to a lower performance. It should be noted, though, that the individual data sets are unique and the contained data itself also impacts the performance. It is also not surprising, that more complex lighting models take more time to compute. However, in all cases, our renderer outperforms the rudimentary volume rendering solution by Ryan Brucks [39], independent of the chosen lighting models. Actually, we achieve real-time performance for VR in all our test cases.

Additionally, we have evaluated the efficiency of our octree implementation for empty space skipping. Figure 6.16 shows a comparison between using the octree and empty space skipping and a variant without them. Both versions were tested using the LAO lighting mode and  $2 \times 2$  SSAA. With the octree, we measured performance improvements for all test data sets of up to 49.4 %. The average improvement was 14.7 %, while the empty space ratio varied between roughly 45 % and 55 %, except for one data set with only 36 % (see Figure 6.17). This shows that our octree

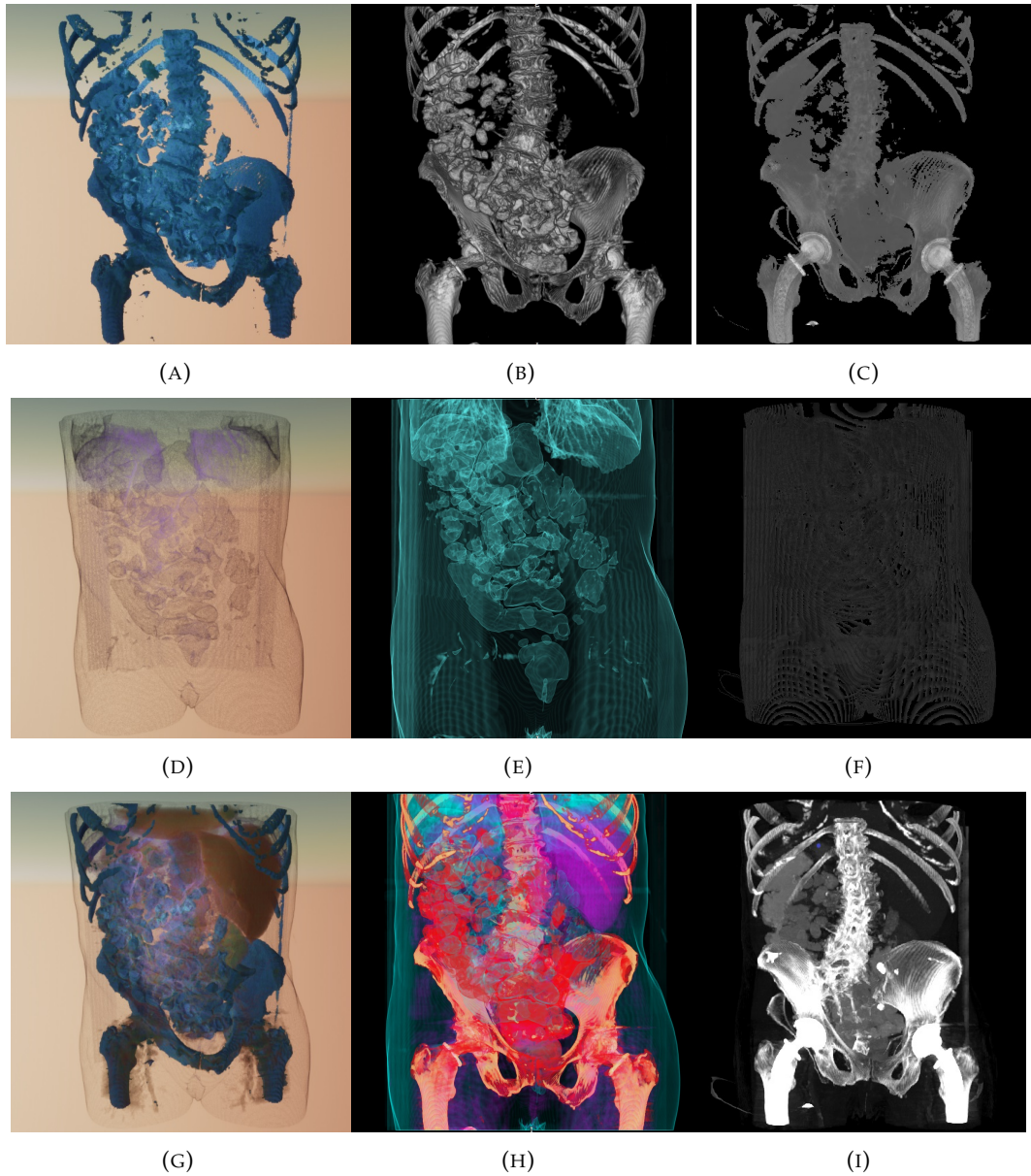


FIGURE 6.12: Comparison of our volume renderer (a,d,g) with the visualization tools RadiAnt using 3D volume rendering (b,e,h) and VTK using maximum intensity projection (c,f,i). Each row shows different windows, from top to bottom: bone, lung and skin, both and soft tissue.

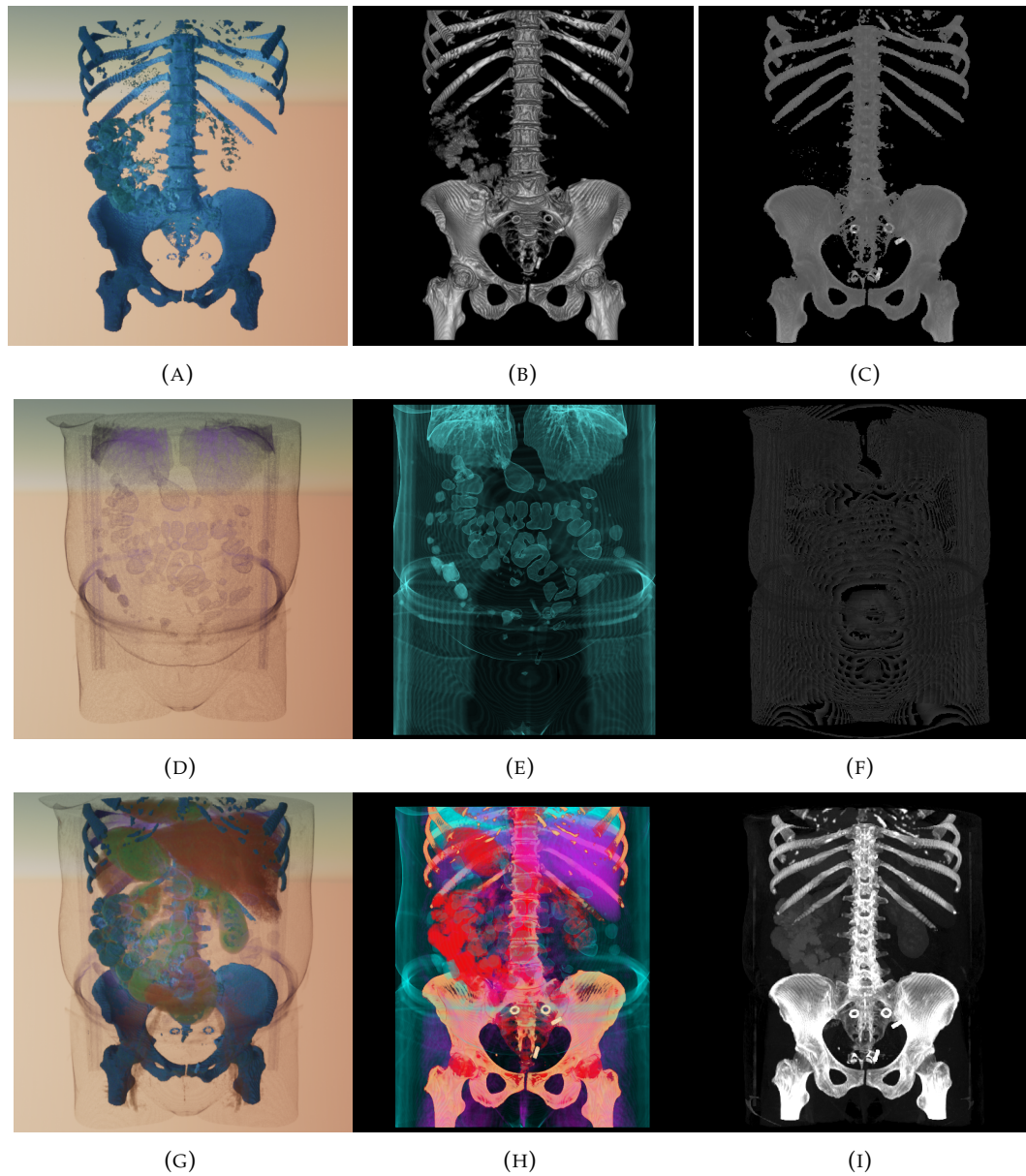


FIGURE 6.13: Another comparison of our volume renderer (a,d,g) with the visualization tools RadiAnt using 3D volume rendering (b,e,h) and VTK using maximum intensity projection (c,f,i). Each row shows different windows, from top to bottom: bone, lung and skin, both and soft tissue.

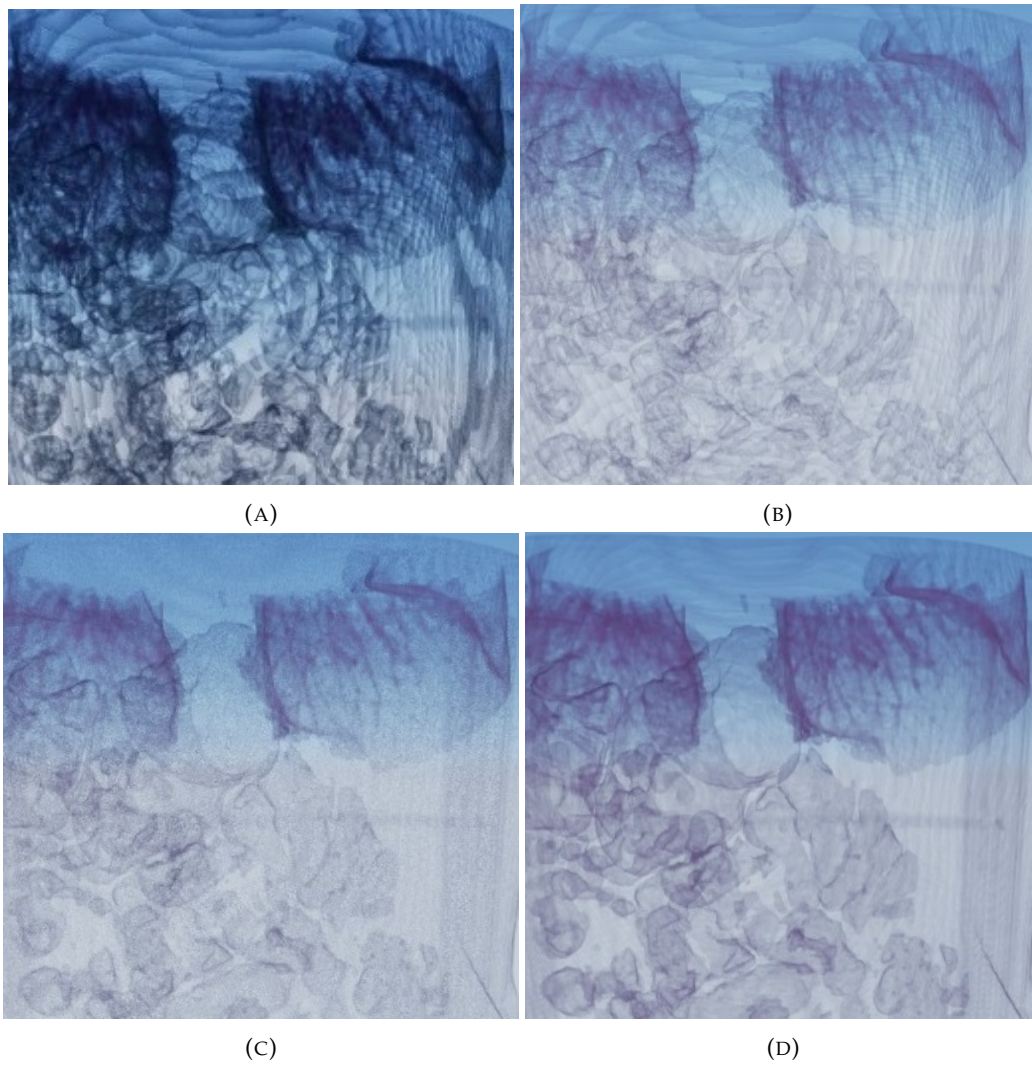


FIGURE 6.14: Artifact minimization using different optimization techniques: (A) no optimization, (B) pre-integration, (C) pre-integration and stochastic jittering, (D) pre-integration and SSAA.



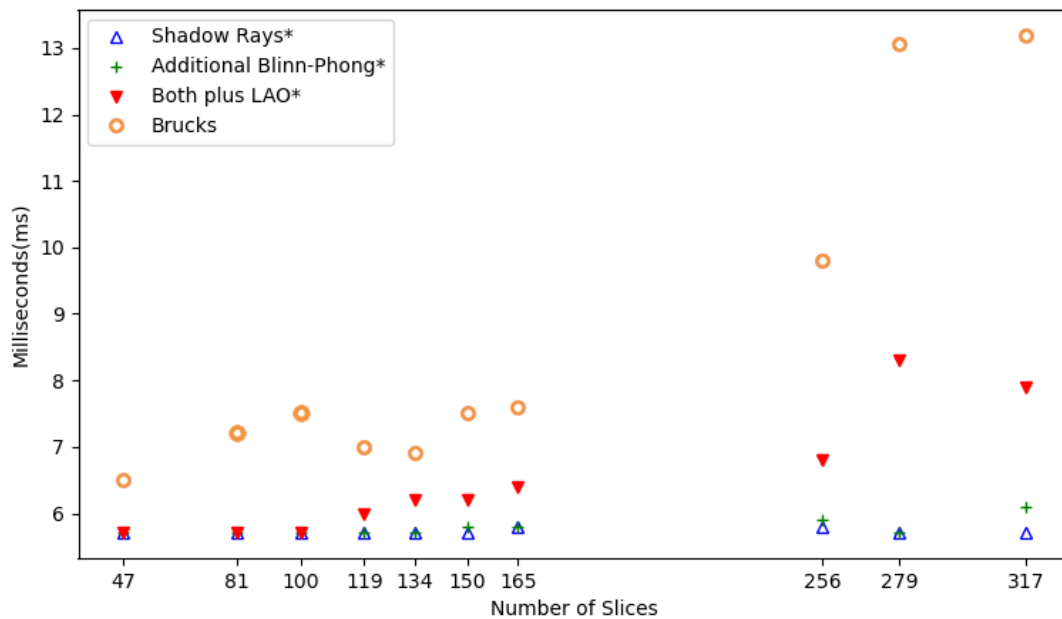


FIGURE 6.15: The Performance using different lighting models and data sets with a varying amount of slices. Our methods are marked with asterisks, “Brucks” is a rudimentary volume rendering integration in Unreal. Even though advanced lighting models increase the computational time, our renderer is real-time capable in all cases and significantly faster than the implementation by Brucks.

implementation is effective in increasing the performance, especially for high-slice data sets. The latter fact may be surprising on the first because the amount of empty, skippable, space is unique to each data set and not necessarily correlated to the number of slices. However, data sets with more slices often do exhibit more potential for empty space skipping because of the higher granularity. This usually leads to a higher sampling frequency and, thus, to a higher number of redundant sampling positions in empty regions.

Finally, we have measured the network performance, specifically the latency. However, in order to get objective and comparable results, we avoided a real internet transmission that is highly dependent on individual factors such as connection quality or distance. Instead, we set up the client and server on two different computers which were connected via a router, and measured the round time of the network messages from client to server and back. The average time was 16.8 ms with a standard deviation of 1.6 ms that is added by our system. Obviously, in the case of an internet connection, additional latency has to be added. To conclude, our application is very well suited for collaborative work as actions from other users are replicated quickly. Accordingly, the user feedback, regarding the multi-user VR experience as well as the medical visualization, is very positive so far.

Tables 6.2 and 6.3, show an overview of all employed datasets, including statistics such as memory consumption, the ratio of empty space, and rendering times in various settings.

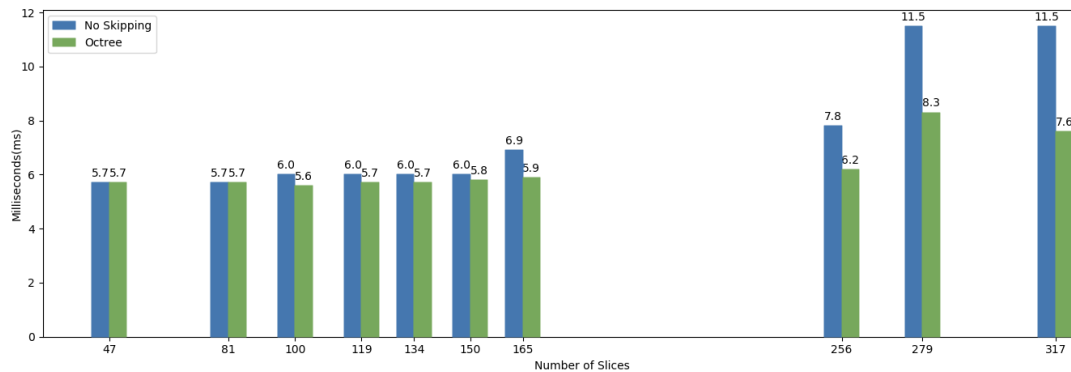


FIGURE 6.16: The performance impact of the octree for empty space skipping over different data sets with a varying number of slices and  $2 \times 2$  SSAA. Especially data sets with a high number of slices benefit the most.

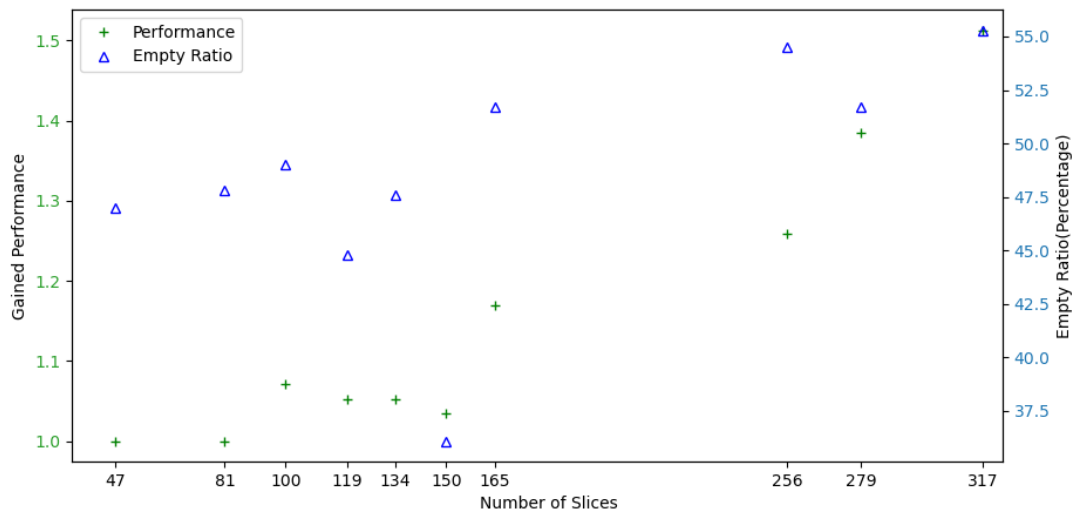


FIGURE 6.17: The performance impact of our octree over different data sets with a varying number of slices and  $2 \times 2$  SSAA, compared to the ratio of empty space of the volume. Data sets with more slices profit the most from the octree, however, the relation seems to be only loosely tied to the ratio of empty space.

TABLE 6.2: Overview over the dataset and their statistics (first half) including the rendering performance in various settings and lighting modes. SM stands for the shadow ray-based lighting, BP for the additional Blinn-Phong lighting, and UE4 for the method by Brucks. The performance measurements were done using early ray termination (ERT).

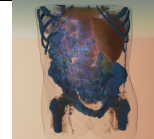
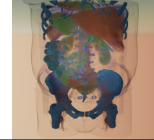
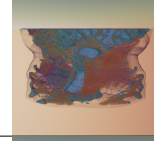
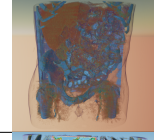
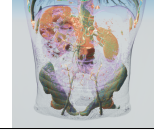
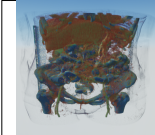
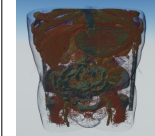
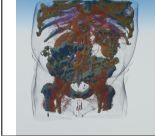
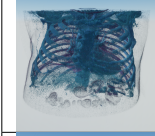
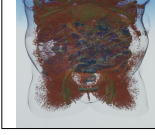
Dataset	Description	Size (8bit)		Empty Ratio	Time in ms (ERT)					
		Images (MB)	Octree (KB)		SSAA+LAO		No SSAA			
					Plain	Octree	SM	BP	LAO	UE4
	p03_78e 512x512x81 ST: 3mm	3.9	30.5	76.1%	5.7	5.7	5.7	5.7	5.7	7.2
	p02_360 512x512x100 ST: 3mm	4.2	60.9	60.9%	6.0	5.6	5.7	5.7	5.7	7.5
	p01_04 512x512x150 ST: 1.5mm	4.5	60.9	40.3%	6.0	5.8	5.7	5.8	6.2	7.6
	p03_788 512x512x256 ST: 1.5mm	6.8	60.9	57.1%	7.8	6.2	5.8	5.9	6.8	9.8
	p08_1b14 512x512x47 ST: 2mm	1.3	76.1	49.0%	5.7	5.7	5.7	5.7	5.7	6.5

TABLE 6.3: Overview over the dataset and their statistics (second half) including the rendering performance in various settings and lighting modes. SM stands for the shadow ray-based lighting, BP for the additional Blinn-Phong lighting, and UE4 for the method by Brucks. The performance measurements were done using early ray termination (ERT).

Dataset	Description	Size (8bit)		Empty Ratio	Time in ms (ERT)					
		Images (MB)	Octree (KB)		SSAA+LAO		No SSAA			
					Plain	Octree	SM	BP	LAO	UE4
	p06_12f9 512x512x134 ST: 3mm	3.0	60.9	51.1%	6.0	5.7	5.7	5.7	6.0	6.9
	p07_16c1 512x512x317 ST: 1.5mm	5.3	76.1	58.5%	11.5	7.6	5.8	6.1	7.9	13.2
	p09_24f 512x512x165 ST: 3mm	3.4	60.9	54.7%	6.9	5.9	5.8	5.8	6.3	7.6
	p09_24d 512x512x119 ST: 3mm	7.7	30.5	48.9%	6.0	5.7	5.7	5.7	6.0	7.0
	p05_f21 512x512x279 ST: 1.5mm	5.4	38.1	54.2%	11.5	8.3	5.7	5.7	8.3	13.1

### 6.1.5 Conclusions and Future Work

We have presented a multi-user virtual reality system for medical visualization based on a state-of-the-art game engine that is capable of 3D visualizing computed tomography data in real time and with high visual quality. This is achieved by our custom ray-marching-based direct volume renderer which we have implemented using shaders and integrated into the Unreal Engine. Our renderer supports different lighting models, transfer function selection, and artifact-reducing methods. Our evaluation shows that we achieve VR-capable framerates of more than 100 Hz even for complex data sets consisting of more than 300 slices and with advanced lighting features such as ambient occlusion enabled. Our system includes a multi-user component and is designed as a shared virtual environment resembling a real operation room, thus, enabling immersive collaborative work between co-located or remote users. Thanks to the combination of the sophisticated game engine, VR, and our fast high-quality direct volume renderer, users can interact with each other and the shared visualized CT data in an immersive virtual environment and (re)view and discuss the 3D data in a comprehensive natural way. This makes our system ideally suited for pre-operative planning, possibly tumor boards, post-operative evaluation, or patient education.

In the future, the interaction possibilities should be expanded with the volume visualization, specifically, it may be beneficial to integrate a dynamic clipping plane for a better view of internal regions and a volumetric drawing tool allowing for quick sketches and annotations inside the volume. Other improvements would be a direct integration and parallelization of the preprocessing part to speed up the workflow and allowing for a dynamic adjustment of the transfer functions. To improve the visualization of complex structures and organs that involve multiple materials support for multi-dimensional transfer functions could be added. Finally, it may be worthwhile investigating if hierarchical irregular grids lead to performance gains over the currently used regular octree.

## 6.2 Anatomy Learning through a Collaborative VR Anatomy Atlas

The last section was concerned with 3D visualization of medical data, such as CT images, in collaborative VR to improve diagnosis, pre-operative planning, etc. Another aspect to consider is how to help with the education of doctors, i.e., how to facilitate efficient learning of anatomical knowledge. Again, specialized VR tools for teaching and training that provide interactive 3D visualizations, such as anatomy atlases, are a promising solution. They can provide highly interactive and engaging learning environments where students can immersively and repeatedly inspect and interact with virtual 3D anatomical structures. Moreover, multi-user VR environments can be employed for collaborative learning, which may enhance the learning experience. Concrete applications are still rare, though, and the effect of collaborative learning in VR – if the collaboration is as beneficial as with traditional learning – has not been adequately explored yet.

In this Section, we, therefore, present a collaborative VR anatomy atlas with an accompanying user study with  $n = 33$  participants that we conducted to evaluate the effectiveness of virtual collaboration on the example of anatomy learning. Specifically, we investigated the learning progress as well as the usability between single-user and multi-user learning.

The work presented in this section is based on our accepted but not yet published paper PP3 in Appendix A.

### 6.2.1 Introduction

The teaching of human anatomy is fundamental in medical education as it forms the basis for the development of clinical and surgical knowledge among professional [340, 183]. Classically, anatomy is thought through dissection, prosection, and lectures. However, dissection is costly and time-consuming, prosection relies heavily on the skill and expertise of the anatomist [101], and lectures may not be effective in promoting active learning and engagement compared to more interactive approaches. Moreover, the availability of human cadavers and animal specimens for dissection is limited [32].

VR has become increasingly prevalent as a tool in (medical) education [236, 304] as it is not affected by these limitations and can provide more intuitive and engaging [343, 17] learning experiences. More (general) examples can be found in Section 2.3. However, most current VR-based learning applications are limited to single-user usage, and there is minimal research on the effectiveness of collaborative VR-based learning. Collaborative learning, in general, has been shown to have positive effects on learning outcomes [50, 165, 194] and to provide numerous other benefits, such as more positive interpersonal relationships [259], though.

To investigate if collaboration in VR (anatomy learning) also provides benefits and more positive outcomes than individual VR learning, we developed a multi-user VR anatomy learning application and conducted a user study to evaluate its effectiveness. Concretely, we examined the participants' learning progress as well as the general usability when using the VR learning environment, individually and in groups. With these results, we provide valuable insights into this sparsely-researched area.

### 6.2.2 Related Work

The use of VR for medical education and training, including anatomy teaching, got much attention lately [286]. For example, Fairen et al. [105] developed and evaluated a VR anatomy teaching tool that provides real-time, interactive 3D representations of various anatomical structures that were augmented with additional information. An evaluation with anatomy students showed very positive results. Codd and Choudhury [72] compared anatomy teaching using 3D virtual reality with traditional methods and, interestingly, found no significant learning advantages using VR. In contrast, Kurul et al. [192] also conducted a study on anatomy training comparing immersive, interactive 3D VR with classical teaching methods and found the former to lead to significantly higher test scores. Another example highlighting the benefits of VR to anatomical education is the Immersive 3D Anatomy Atlas by Gloy et al. [131]. It provides a realistic 3D model of the human body in an immersive environment and allows users to explore individual anatomical structures interactively. An evaluation showed that the VR group took significantly less time to answer anatomical questions and had significantly better test results than students that learned using textbooks.

Only a few works allow for collaboration in VR and even fewer investigate its effects and benefits, though. For instance, Boedecker/Schott et al. [29, 323] developed a collaborative, immersive VR application for liver surgical planning that provides multiple teaching and training scenarios for varying group sizes. An exploratory

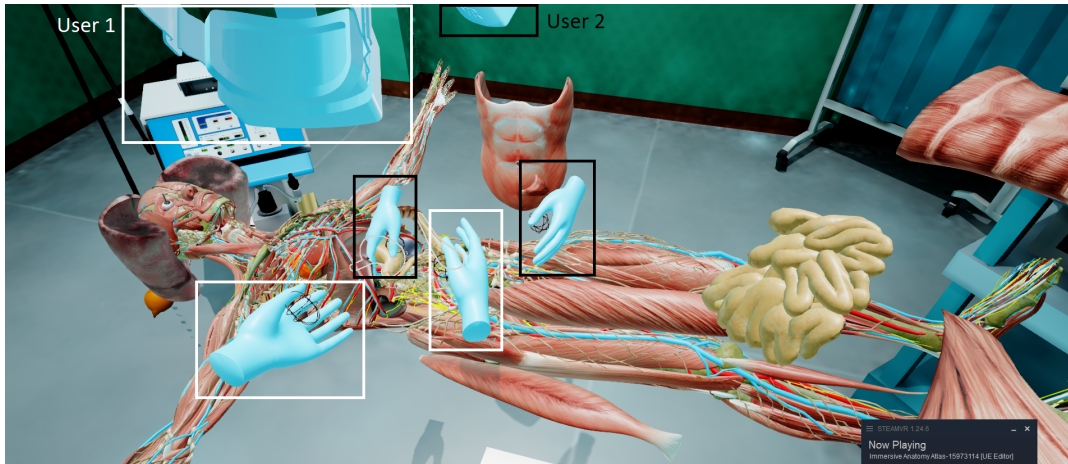


FIGURE 6.18: Two users within our Collaborative VR Anatomy Atlas examining anatomy. Each user has an avatar consisting of a virtual HMD and one pair of hands (light blue, highlighted in black and white boxes).

study indicated positive outcomes for usability and presence. Similarly, De Back et al. [78, 12] also presented a collaborative, immersive VR learning environment and found that collaborative learning provided greater learning gains compared to conventional textbook learning. For a more detailed overview and review of VR for anatomy education, we refer to the work by Lee et al. [205].

### 6.2.3 Implementation

For our work, we decided to use the Immersive Anatomy Atlas by Gloy et al. [131] as a basis, as it already provided a good implementation of a VR anatomy learning application. For our Collaborative VR Anatomy Atlas, we extended it mainly by multi-user functionality in order to allow multiple users to meet, interact, and collaboratively learn within a shared environment, see Figure 6.18. Each user is represented by an avatar consisting of a virtual HMD and a pair of hands with which they are able to grab, move, and interact with objects. We decided on this avatar model, as it doesn't require complicated scanning setups and is not prone to distracting or glitchy behavior. We use a client-server model based on the network functionality provided by the Unreal Engine 4, which allows for shared learning sessions between users in the same local network or over the internet. The avatars, body parts, and other interactive objects, such as the operation table, instruments, and tablets, get replicated (synchronized) between users using remote procedure calls. The network traffic is minimized using struct replication, delta replication, caching, and careful selection of reliable/unreliable replication channels. We also developed a VR quiz (post-test) to evaluate participants' anatomy knowledge after their learning session.

### 6.2.4 Study

For our study, we formulated the following research questions that we intend to answer: ( $R_1$ ) Is the Collaborative VR Anatomy Atlas effective for anatomy learning? Based on prior research that found benefits in collaborative learning [196, 259], we wanted to investigate if ( $R_2$ ) collaborative learning in VR also leads to better learning

outcomes than individual VR learning. Additionally, we wanted to evaluate the usability ( $R_3$ ), in general, and especially if there are any differences between individual and collaborative learning.

### 6.2.5 Design and Setup

For our study, we employed a between-subject design, hence, we divided the participants randomly into two groups: one group testing the single-user condition and the other group testing the multi-user condition (pair-wise, same room).

During the learning sessions using our Collaborative VR Anatomy Atlas, the participants were represented through three-point tracked avatars and were able to freely move around using room-scale VR and teleportation. The virtual environment resembled an operating room and included a virtual anatomic 3D model that they were supposed to interact with and explore in order to learn about the anatomy. To provide a good user experience, we ensured that the frame rate was maintained at 90 frames per second.

In order to evaluate the learning effectiveness, we designed a multiple-choice test consisting of 8 general anatomy questions. This test was conducted two times: one time before the learning session on paper (pre-test), and one time after the learning session directly in VR through our VR quiz (post-test). During the quiz, the participants had the opportunity to see and learn the correct answers even after answering incorrectly. By comparing the results of the two tests, we calculate the learning progress. Additionally, we employed the System Usability Scale (SUS) [37] to test the usability.

#### Procedure

After giving consent, the participants were asked to complete a demographical questionnaire and complete our pre-test questionnaire (on paper). Following this, the Collaborative VR Anatomy Atlas application and its usage were briefly explained. Lastly, the participants had time to freely explore and familiarize themselves with the VR environment.

During the following learning session, the participants had to explore (individually, or team-wise) the virtual anatomic model and complete various tasks. For instance, discovering the human anatomy, searching for specific organs (e.g., the spleen, pancreas, liver), and finding answers to the pre-test questions. No assistance was given during task completion, but the tasks were repeatable and no time limit was given.

Upon completion of the tasks, the participants were transitioned to the quiz level and took the anatomy post-test. There, they had to answer the shown questions by pressing the corresponding 3D buttons. After the post-test, the participants had to complete the usability questionnaire (on paper). They were also asked if they experienced any motion sickness and to provide subjective feedback. The procedure was identical for both conditions, with the exception that the participants of the multi-user group were explicitly instructed to work together on the anatomical tasks and to learn collaboratively. However, at the VR quiz level, they were required to complete the post-test independently.



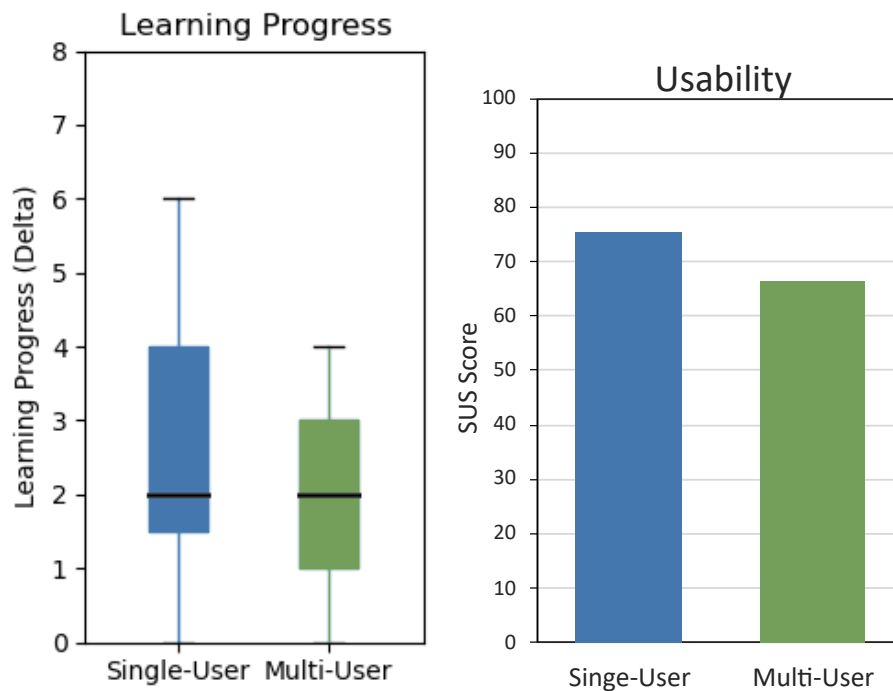


FIGURE 6.19: Left: Learning progress (delta between pre- and post-test) for single- and multi-user groups. The single-user group learned, on average, slightly better. Right: The SUS scores. The scores are generally high but the single-user group's feedback is more positive

### 6.2.6 Results

As the data was, as expected, normally distributed, we conducted independent samples t-tests to test for statistically significant differences between the single and multi-user groups.

The study was conducted with  $n = 33$  participants (11 single user, 22 multi-user). The single-user group was made up of 2 female (18.2 %) and 9 male (81.8 %) participants while the multi-user group was made up of 14 men (63.6 %) and 8 women (36.4 %). Moreover, a substantial percentage of single users (54.5 %) and a smaller percentage of multi-users (22.7 %) reported having extensive experience with VR, a significant percentage of single users (36.4 %) and multi-users (31.8 %) reported having used VR before, while a minority of single users (9.1 %) and a substantial percentage of multi-users (45.5 %) had no experience with VR.

In order to investigate the learning effectiveness, we computed the participants' learning progress as the difference (delta) between the pre- and post-test results, see Fig. 6.19 (left). The single-user group, on average, did have slightly higher learning progress: the mean score was 2.636 ( $SD = 1.859$ ), whereas the multi-user group's mean score was 1.818 ( $SD = 1.140$ ). The median, however, is more similar between the groups. A t-test showed a marginal difference in the mean scores between the groups: ( $t(31) = 1.569, p = 0.127$ ). However, the result is still above the usual threshold of  $p \leq 0.05$  for statistical significance.

The perceived usability of the Collaborative VR Anatomy Atlas was measured using the SUS. The corresponding SUS scores were calculated using the standard methodology and are depicted in Fig. 6.19 (right). Overall, the participants provided

positive feedback and moderate to high ratings. The mean SUS score for the single-user group was 75.227 ( $SD = 8.976$ ) and for the multi-user group 66.364 ( $SD = 14.15$ ). The t-test revealed that there is a noticeable difference in means between the single-user and multi-user groups, although the usual threshold of  $p = 0.005$  for statistical significance was just not reached ( $t(31) = 1.887, p = 0.069$ ).

### 6.2.7 Discussion

Looking at the results, participants in both conditions had high learning progress. These positive results may come due to the VR learning environment allowing the participants to interact with the content in an immersive, engaging, and interactive way, or the 3D representation, which could have been helpful to understand the subject matter and the spatial relations between anatomical structures. We can therefore answer the first research question  $R_1$ : our Collaborative VR Anatomy Atlas is, generally, effective in enhancing the knowledge and understanding of anatomy, which is in line with prior research [236, 304], that found learning using VR to be beneficial.

Interestingly, the learning progress is not higher for the collaborative learning condition. In fact, it is slightly (but not statistically significant) lower. Thus, we could not find VR learning to be more effective in collaboration than individually, which answers the research question  $R_2$ . This result is surprising since collaborative learning is generally considered beneficial [50, 165, 194]. A potential explanation could be the single-user group having less prior knowledge about anatomy, thus, having more learning potential. Another possible reason may be that the participants that learned individually could better focus on the task while participants in the shared environment were more distracted by each other and the more complex multi-user environment, leading to a higher cognitive load. Additionally, our chosen avatar representation may have not provided a sufficient level of immersion, personalization, and embodiment, which possibly lead to a low feeling of social presence. One also has to consider the option that the task of anatomy learning in VR may be one that is not benefiting from collaboration.

Regarding our research question  $R_3$ , we found that the results of the usability questionnaire are positive, especially for the single-user group. This shows that our Collaborative VR Anatomy Atlas provides a medium to good user experience. The score for the multi-user group is noticeably lower, though. This reinforces our assumption that participants in the multi-user condition perceived the environment as more complex and demanding, potentially leading to a higher cognitive load. Thus, the low usability may be a core explanation for the lower learning progress in the multi-user condition. The subjective feedback given by the participants during and after the learning session was generally very positive.

### 6.2.8 Conclusions and Future Work

In order to investigate the effectiveness of collaborative learning in VR, we developed the Collaborative VR Anatomy Atlas, a virtual reality system for anatomy education, and evaluated it by conducting a user study with  $n = 33$  participants in which we compared individual with collaborative learning. Our application provides an immersive multi-user learning environment in which users can interactively explore detailed anatomical structures. The results show that our Collaborative VR Anatomy Atlas was effective in anatomy learning for both single and multi-user scenarios. Moreover, the participants reported medium to high usability

scores. However, we could not find significant advantages (or differences) regarding the learning effectiveness of the collaborative learning scenario. The usability even tended to be slightly lower. We suspect this to be due to the more complex shared environment and a higher cognitive load. Other reasons could be that the used avatars were not immersive enough, leading to low social presence, or that learning anatomy in VR is a task that does not necessarily benefit from collaboration. Nonetheless, we demonstrated that collaborative VR can be an effective approach to anatomy learning with huge potential to improve the current medical education systems.

In the future, we see much potential in further enhancing the usability and user experience for multi-user usage and would recommend to conduct further studies to examine the impact of presence, cognitive load, and more immersive avatars. It would also be important to normalize the learning progress based on pre-existing knowledge.



## Chapter 7

# Conclusions and Outlook

This chapter provides a summary of this thesis and its key contributions (Section 7.1). Additionally, a brief outlook on promising future research areas is given (Section 7.2). To keep this chapter brief, however, only the main concepts and results get presented here; more detailed results, including discussions, can be found in the respective chapters.

### 7.1 Summary

Multi-user VR is an upcoming technology with immense potential to raise the quality of remote collaborative work to a new level throughout the industry, entertainment, and research. The benefits are a more immersive 3D visualization, natural user interactions, and the ability to provide arbitrary, highly detailed, virtual scenes, objects, and scenarios that can be freely, interactively, and repeatedly explored. These can also be augmented with live-captured data and avatars of the users. Moreover, the shared environment is accessible from anywhere to any number of participants that can interact and collaborate with each other. These aspects make multi-user VR an ideal tool for training and education, data visualization, telepresence/remote assistance, and a good virtual testbed for simulations. The goal of this thesis was to address various issues and challenges that hinder multi-user VR applications from fully using their potential in practice. These challenges were mostly related to the reconstruction, visualization, and rendering of 2D and 3D data (i.e., avatars, live-captured RGB-D images/point clouds, and mesh-based environments) in the shared virtual environment. Furthermore, we considered closely related tasks such as compression and streaming of this data. While each contribution's section already entails a closely related conclusion, we want to give a brief recap of them here, outline the main findings, and draw a final résumé.

The first contribution to multi-user VR and telepresence is the development and evaluation of a low-latency point cloud streaming and rendering pipeline that we embedded in the application domain of surgery assistance. Our pipeline allows remote experts to observe live operations in a virtual operating room and interact with the local doctor and other remote participants. This is achieved by using multiple RGB-D cameras and real-time processing, streaming, and reconstructing of the captured data. Each user, also remote ones, gets visualized by live-reconstructed point cloud-based avatars. Most telepresence systems only support one RGB-D camera, are complicated to set up, or create high latency and bandwidth. In contrast, our system is easy to use, is designed for minimal latencies, and supports an arbitrary number of RGB-D cameras. Using custom filtering and compression algorithms, parallelization, and two self-developed fast rendering methods, we achieve motion-to-photon latencies of just 120-150 ms and 23.4 MB/s of bandwidth per camera (lossless). We also developed a prototype for real-time face reconstruction that allows one

to see other users' faces, including the current mimics, even though they are wearing HMDs. Our user study, which we have conducted in a hospital, showed that the system is perceived very well by the doctors and they have a high interest in using it in practice. Also, they had strong feelings of presence, thanks to, among other things, the live-reconstructed 3D avatars and our custom point cloud rendering solutions.

In VR-based telepresence systems such as the one we proposed, live reconstructions of the remote scenes and avatars are usually based on RGB-D data, however, these require a lot of bandwidth when streamed raw to other participants. Especially the depth data poses an issue, as it cannot be compressed effectively and artifact-free using the standard color image and video compression algorithms. Time constraints are another challenge, as the images usually get captured with 30 Hz and any processing and compression should be performed equally fast or even faster. We, therefore, aimed to further enhance the real-time data streaming between participants in collaborative VR by presenting a lossless real-time compression algorithm for depth data. It achieves a higher compression ratio than current algorithms such as RVL and helps to reduce the bandwidth requirements significantly, even though we focused on achieving a good trade-off between speed and compression ratio instead of absolutely minimal sizes. Our algorithm is based on the RVL algorithm and extends it by our self-developed span-based adaptive intra-image predictor, inter-image delta computations, parallelization, and further compression using Zstandard. An evaluation showed that our algorithm achieves significantly higher compression ratios than RVL and other compression methods while still being real-time capable.

In order to provide detailed, high-quality live reconstructions in multi-user VR, the captured depth images not only have to be effectively compressed but also pre-processed. Raw depth images inherently contain noise and, perhaps more critically, are usually incomplete, meaning, riddled with areas of no or invalid data. We tackled the latter issue by proposing a deep-learning-based approach to depth image inpainting/completion, which is able to quickly reconstruct the missing areas without any color image guidance. For this, we adopted two U-Net-based networks that (like most networks) originally were designed for color-image inpainting. The first network uses partial convolution layers and the second one is a patch-based GAN. Using multiple public and custom-made data sets, our comprehensive evaluation showed significant improvements in image quality. The GAN-based network performed best on images with smaller hole-to-image ratios while the network with partial convolutions performed uniformly well. Moreover, the inference timings for both networks were low enough for real-time employment. More sophisticated networks such as LaMa do provide, on average, better results but are significantly slower and not quite real-time capable anymore.

Traveling greater distances in virtual environments is often done using the teleport locomotion metaphor as, among other reasons, it is safe regarding cybersickness. In shared virtual environments, such as our previously presented telepresence system, this can lead to confusion and a reduction in presence for onlookers as they get no visual feedback about the process. In order to preserve the feeling of presence when teleporting, we developed and evaluated suitable teleport visualization methods that can be shown to onlookers while teleporting. These included continuous and non-continuous ones such as particle effects, portals, walking animations, and others. Our study with multiple scenarios and locomotion durations showed that the visualizations can help significantly with the comprehension of the deliberate teleport process. Specifically, we found benefits regarding the social- and spatial presence, spatial awareness, and prevention of confusion. However, we found also that the benefits of the visualizations decrease when increasing the locomotion

speed/shortening its duration, and that type of visualization is highly relevant and not all visualizations perform equally well. Our results indicate that continuous ones often tend to perform better and that especially the realistic walking animation consistently performed best.

As discussed previously, having an immersive, detailed virtual environment is an important aspect in multi-user VR. Some domains, for instance, robotics, the automotive industry as well as the aerospace industry, often are especially dependent on large-high-quality 3D environments. One example is virtual testbeds, which require realistic, feature-rich large-scale landscapes to simulate the interaction and behavior of vehicles or other systems with the local terrain. To also contribute to this area, we presented multiple novel procedural terrain generation systems that focus on a series of complex issues. First, we proposed a PTG system that is able to generate huge, varied landscapes with plausible biome distributions. To achieve this, various procedural methods, exemplar-based methods, and a simplified climate simulation were combined into a single pipeline that also is easy to use. Next, we also presented a PTG approach to generate landscapes with realistic-looking and plausibly distributed water bodies, such as rivers and lakes, which is a sparsely researched but nonetheless important topic. Our design is based on artificially created drainage basins that dictate the eventual terrain. Our evaluations show that the proposed PTG systems are not only easy and quick to use but also generate huge, highly variable terrains that are able to closely mimic realistic terrains. Finally, we theoretically discussed a system that, given an input DEM, would produce visually and statistically similar look-alikes. This would enable engineers and researchers to quickly generate a number of slightly varying terrains that, at the same time, follow given requirements. This, in turn, would help to virtually test their systems on a vast number of conditions.

Exploring potential applications domains for multi-user VR, we found the medical area to be especially promising. We already found that having highly detailed virtual environments and live 3D reconstructions of remote scenes and other participants are important for multi-user VR and telepresence. However, in the medical area, there is often also additional medial data that could be 3D visualized for collaborative, interactive inspection in VR, e.g., to help with diagnosis or pre-operative planning. Thus, we developed a custom direct volume renderer for medical data and integrated it into a multi-user VR application that is based on the Unreal Engine 4. This allows doctors to collaboratively and remotely visualize and interact with volumetric medical data, e.g., CT scans, in an immersive and more natural way than the currently used programs with standard 2D interfaces. Our renderer provides various advanced lighting techniques and dynamically selectable transfer functions, as well as various optimization techniques that increase performance and reduce artifacts. For instance, pre-integration, stochastic jittering, and an octree. Thus, the evaluation showed that our renderer provides high-quality visuals and VR-capable framerates of more than 100 Hz, which makes it ideal for tasks such as pre-operative planning or for spontaneous consultation during VR-assisted operations.

Another highly relevant area in medicine in which (multi-user) VR can provide benefits is education, e.g., with learning of anatomical knowledge through so-called VR anatomy atlases. In classical learning, collaboration tends to be beneficial. If this is also the case when using VR, is hardly researched yet. Thus, to evaluate the effectiveness of collaborative learning in VR and its potential benefits over individual learning, we developed a multi-user VR anatomy learning application that provides users with the ability to collaboratively explore and interact with a virtual 3D model

of the human anatomy and its individual organs. A user study revealed that the application is effective for anatomy learning and provides good usability. However, we could not find collaborative learning to be more effective than individual learning.

With the contributions made within this thesis, we tackled many challenges and significantly raised the current state of the art of collaborative VR throughout a wide range of areas. For instance, our contributions regarding real-time RGB-D data enhancement, streaming, 3D reconstruction, and rendering substantially improve the ability to provide highly detailed live 3D data and avatars to local and remote participants in the virtual environment, i.e., in the context of a telepresence/surgery assistance system. This increases the quality of these systems and makes them more viable and accessible in practice. Using our volume renderer, the participants are also able to immersively visualize and interact with volumetric data, such as CT scans, directly in the shared virtual environment and our proposed teleport visualizations help to prevent confusion and retain presence for users in shared environments. Lastly, our contributions to PTG improve the capabilities regarding realistic, detailed environment visualization and simulation, i.e., in virtual testbeds. Eventually, with the ensemble of contributions that we presented within this thesis, including not only novel algorithms and methods but also studies and comprehensive evaluations, we were able to improve collaborative VR on many fronts and provide critical insights into various research topics.

What may come in the future, or at least what we see as worthwhile future research directions, will be outlined in the following.

## 7.2 Outlook

Despite the recent advances in collaborative VR and the contributions of this thesis, there are still many areas of future research that have great potential to improve the current state of collaborative VR even further. In the following, we want to show a brief outlook on these possible directions and promising research ideas. For specific limitations of our contributions and more detailed ideas to mitigate them, we refer to the individual contributions' future work sections.

As mentioned, high-quality personalized avatars are crucial for collaborative VR, and in this regard, we see still a lot of potential for improvement regarding (re)construction and rendering. One prospective way for future research would be to develop algorithms that not only reconstruct avatars from RGB-D data but also smoothly and more efficiently merge the data from multiple views to prevent occlusions and reduce the uncertainty inherent in the data. Current algorithms that achieve this are computationally very complex and take too much time. Related is the issue of the peoples' faces being occluded by the HMD. As seeing the other person's face is highly relevant and beneficial for many applications, it is highly important to investigate methods to reconstruct the faces, including the mimics, in real time. Fortunately, modern HMDs start to include eye- and facial trackers more often, which helps to get live data about the mimics and can be used for facial reconstruction. To achieve good results that accurately depict the real person's face and don't look weird or extremely unnatural is not trivial though. Our prototype may be a good step in the right direction.

We also see tremendous potential in taking advantage of the recent developments in deep learning. Training and employing specialized CNNs showed to be highly beneficial in various areas, mostly image processing and computer vision, but also in rendering. Collaborative VR could greatly benefit from these developments.



For instance, we think deep learning and CNNs have great potential in RGB-D processing tasks such as denoising and inpainting, but also super sampling; the limited depth sensor resolution is actually a significant obstacle. Moreover, recent works showed that deep learning can also be successfully used for 3D rendering. Therefore, it might be worth it to see if and how neural networks can be used to replace traditional RGB-D reconstruction and rendering algorithms. The same questions apply to direct volume rendering in VR. One approach could be to train a neural network to generate highly realistic avatars based on the current pose/skeleton, including mimics provided by eye trackers, etc., and pre-made scans of the user. Naturally, ease of use and high usability are important factors for practicability. Therefore, research and development would be needed to track the user's pose/skeleton just using a consumer webcam or smartphone and create the scans, too, just using a couple of images or a short video by a typical smartphone camera.

Despite our contribution to RGB-D compression, there is still potential for further improvement, as, still, huge amounts of data have to be transmitted over often limited connections. The most profitable approach may be to try to make use of the GPU and develop effective compression algorithms that are highly parallelizable, even though this is traditionally difficult, as the potential for efficiency improvements is huge. Therefore, it could also be an interesting approach to take advantage of the recent advances in deep learning and try to develop and employ state-of-the-art models for deep data compression. An inherent advantage would be that the inference is usually taking place on the GPU. The issue could also be tackled from another angle, though. Namely, by trying to find more efficient solutions on a higher abstraction level. Here, techniques from related research fields such as networking get relevant. An idea could be to develop adaptive rendering or compression strategies that dynamically change the rendering/compression quality/precision based on the available bandwidth, similar to a classical LOD system. Using eye tracking, the quality could also be adapted spatially, i.e., only the areas that are looked at get rendered/compressed in the highest possible quality. Another avenue to increase the efficiency would be to consider predictive rendering techniques. On the one hand, these could be applied to the rendering of the final images themselves to save rendering time, but on the other hand, the avatar's motion could be predicted, too, to lower the required update frequency (rendering, transmission). Moreover, the perceived latency for the own avatar could possibly be reduced.

There are even more possible research topics regarding avatars, though. For instance, we discussed how the avatar representation and their movement through the scene are highly important for the sensation of (tele)presence. Our research into teleport visualizations definitely helps to improve the depiction for onlookers, however, it is still necessary to also consider the visualizations' effects on the acting user himself, i.e., usability, embodiment, and presence, especially as they may consume a significant amount of time. Finding a good trade-off between the conflicting requirements between onlookers and the acting user would be an interesting challenge.

In the end, we can agree that (multi-user) VR is an exciting and highly relevant technology, that, still, has huge growth potential and exhibits various interesting directions for future research.



## Appendix A

# Publications

Some parts of this thesis appeared already in the following core publications (chronological order):

- PC1 Fischer, R., Boeckers, J., Zachmann G. (2022). Procedural Generation of Landscapes with Water Bodies Using Artificial Drainage Basins. In: Computer Graphics International (CGI). Lecture Notes in Computer Science, vol 13443. Springer.
- PC2 Fischer, R., Mühlenbrock, A., Kulapichitr, F., Uslar, V.N., Weyhe, D., Zachmann, G. (2022). Evaluation of Point Cloud Streaming and Rendering for VR-Based Telepresence in the OR. In: Virtual Reality and Mixed Reality. EuroXR 2022. Lecture Notes in Computer Science, vol 13484. Springer.
- PC3 Fischer, R., Chang, KC., Weller, R., Zachmann, G. (2020). Volumetric Medical Data Visualization for Collaborative VR Environments. In: Virtual Reality and Augmented Reality. EuroVR 2020. Lecture Notes in Computer Science, vol 12499. Springer
- PC4 Fischer, R., Dittmann, P., Weller, R., Zachmann, G. (2020). AutoBiomes: procedural generation of multi-biome landscapes. In: The Visual Computer (36).
- PC5 Fischer, R., Dittmann, P., Schröder-Dering, C., Zachmann, G. (2020). Improved Lossless Depth Image Compression. In: Journal of WSCG, 28.

Currently submitted works that include parts of this thesis (acceptance still pending (PP2) or accepted but not published yet (PP1,PP3)):

- PP1 Fischer, R., Jochens, M., Weller, R., Zachmann, G. (2023). How Observers Perceive Teleport Visualizations in Virtual Environments. In: ACM Symposium on Spatial User Interaction (SUI).
- PP2 Fischer, R., Roßkamp, J., Hudcovic, T., Schlegel, A., Zachmann G. (2023). Inpainting of Depth Images using Deep Neural Networks for Real-Time Applications. In: International Symposium on Visual Computing (ISVC).
- PP3 Almaree, H., Fischer, R., Weller, R., Zachmann, G. (2023). Collaborative VR Anatomy Atlas - Investigating Multi-user Anatomy Learning. In: Virtual Reality and Mixed Reality. EuroXR 2023.

Moreover, there are a number of co-authored papers that are related and where I was involved by giving ideas and advice, helped with the evaluation or the writing. However, they are not directly part of this thesis. Again, in chronological order:

- PR1 Mühlenbrock, A., Fischer, R., Schröder-Dering, C. et al. (2022). Fast, accurate and robust registration of multiple depth sensors without need for RGB and IR images. In: *The Visual Computer*.
- PR2 Mühlenbrock, A., Fischer, R., Weller, R., Zachmann, G. (2021). Fast and Robust Registration of multiple Depth-Sensors and Virtual Worlds. In: *International Conference on Cyberworlds (CW)*, pp. 41-48.
- PR3 Mühlenbrock, A., Fischer, R., Weller, R., Zachmann, G. (2021). Fast and Robust Registration and Calibration of Depth-Only Sensors. In: *Eurographics 2021 Posters*.
- PR4 Reinschluessel, A., Fischer, R., Schumann, C., Uslar, V., Muender, T. et al. (2019). Introducing Virtual & 3D-Printed Models for Improved Collaboration in Surgery. In: *Proceedings of the 18th Annual Meeting of the German Society for Computer- and Robot-Assisted Surgery (CURAC)*.

## Appendix B

# Additional Plots for Section 3.2

In the following, I depict all plots that were generated during the evaluations for Section 3.2 – How Observers Perceive Teleport Visualizations in Virtual Environments. They were not all directly shown in the mentioned section to maintain clarity.

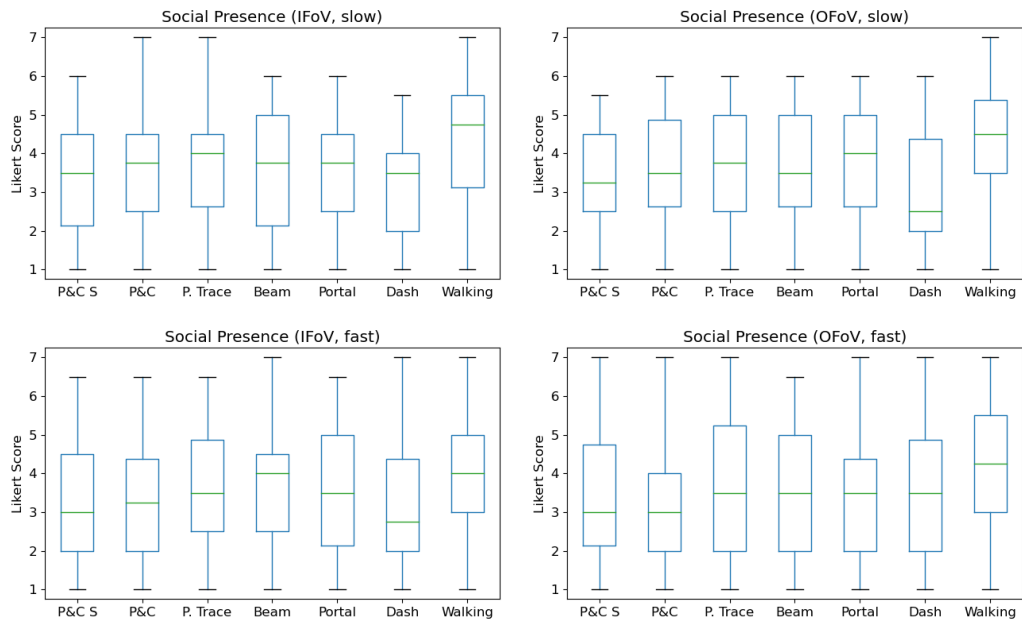


FIGURE B.1: Social presence in the IFoV/OFoV scenarios and slow/fast experiment variants.

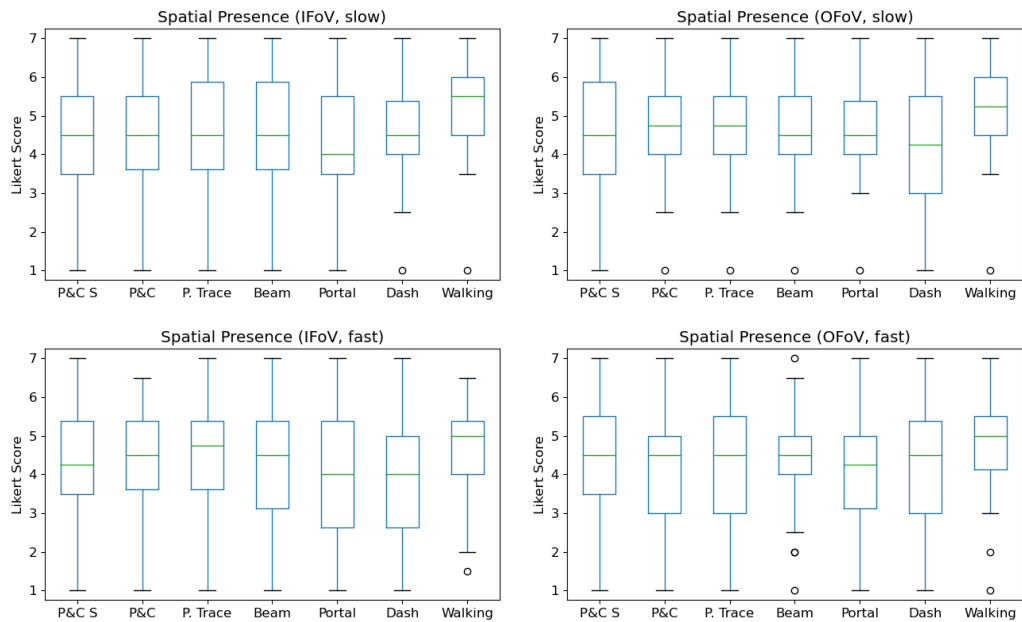


FIGURE B.2: Spatial presence in the IFoV/OFoV scenarios and slow/fast experiment variants.

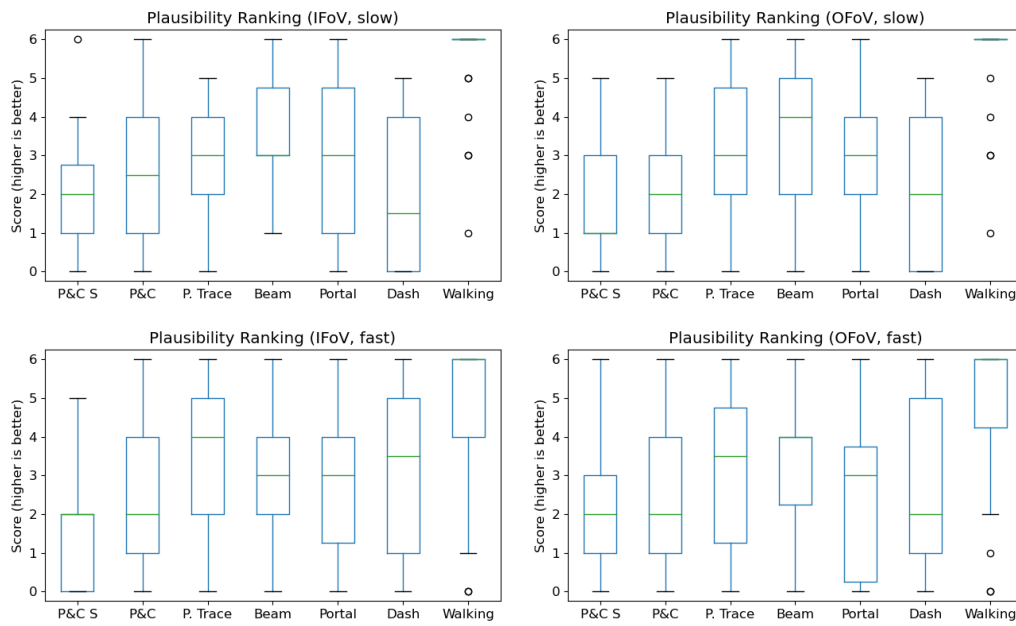


FIGURE B.3: Plausibility in the IFOV/OFoV scenarios and slow/fast experiment variants.

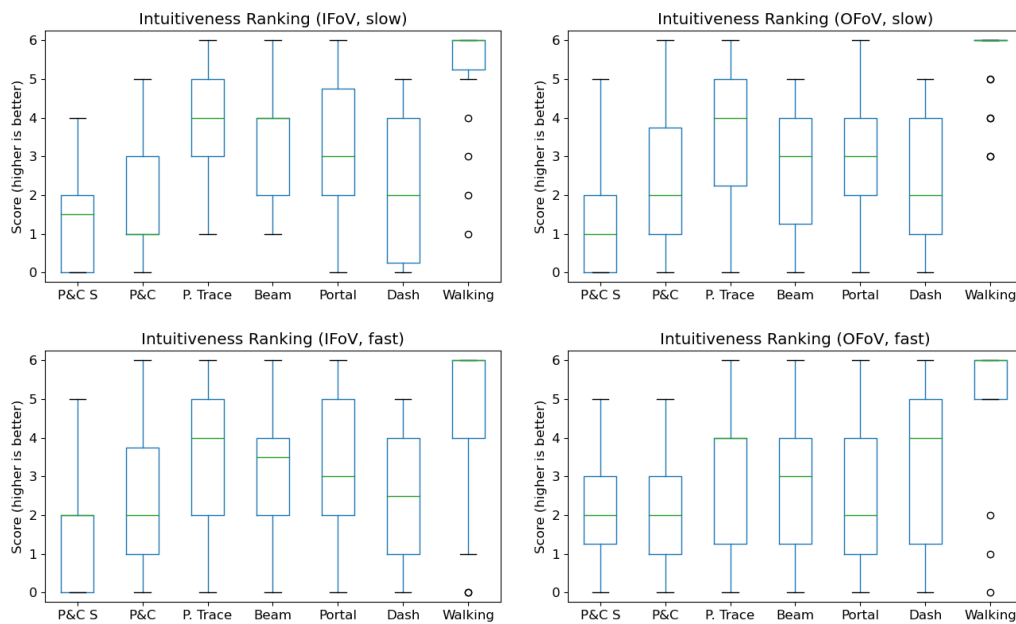


FIGURE B.4: Intuitiveness in the IFOV/OFoV scenarios and slow/fast experiment variants.

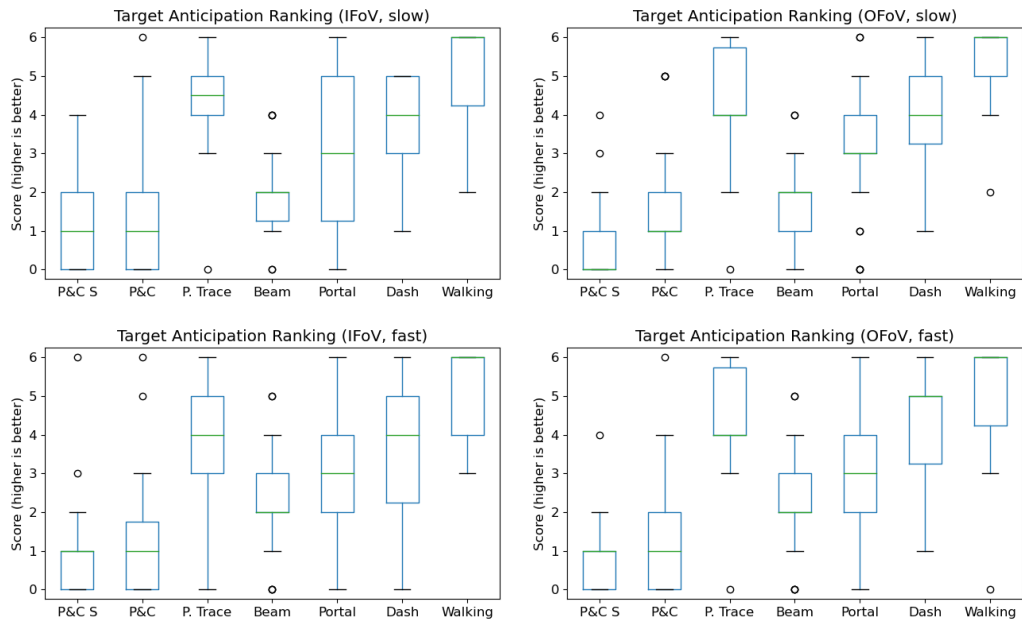


FIGURE B.5: Target anticipation in the IFOV/OFOV scenarios and slow/fast experiment variants.

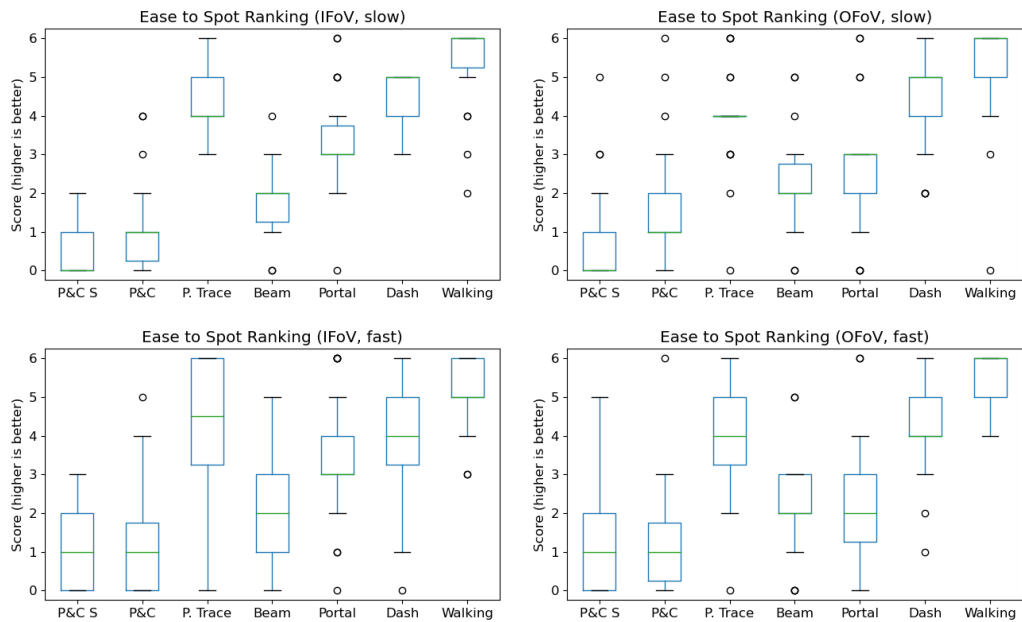


FIGURE B.6: Ease to re-spot in the IFOV/OFOV scenarios and slow/fast experiment variants.



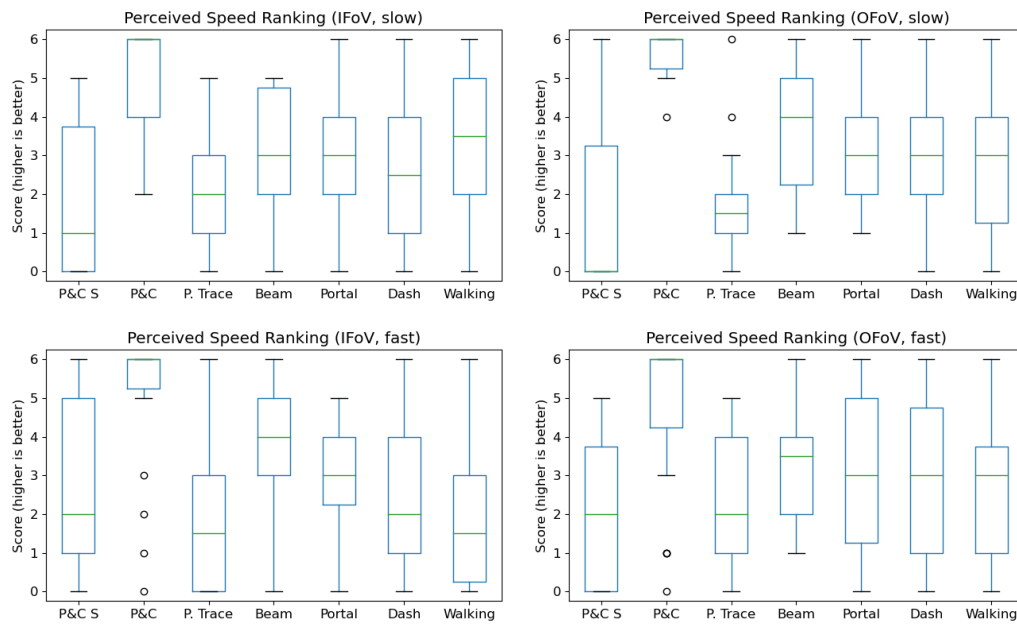


FIGURE B.7: Perceived speed in the IFOV/OFOV scenarios and slow/fast experiment variants.

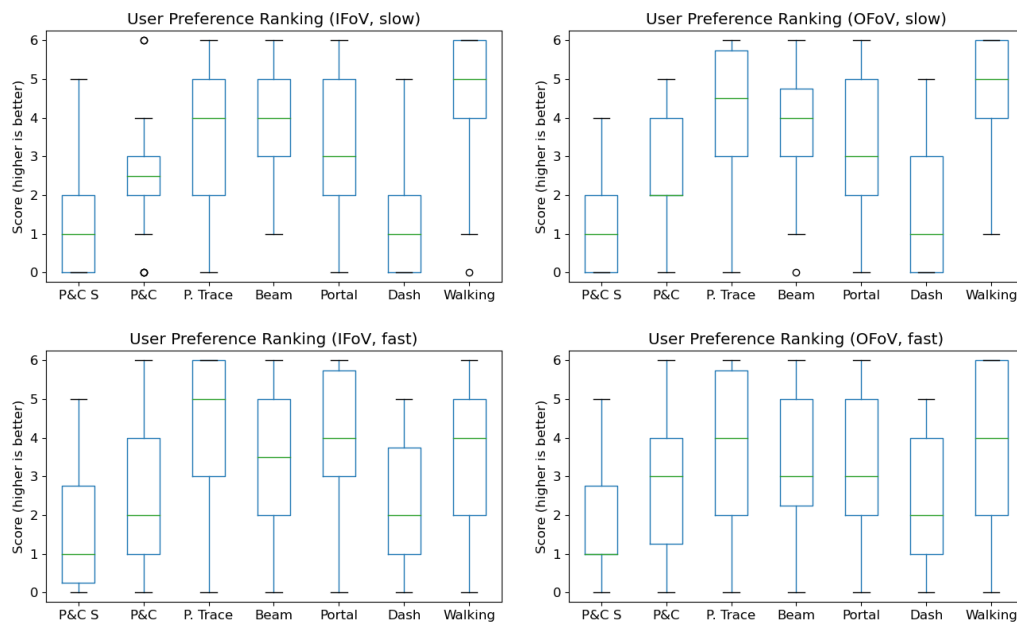


FIGURE B.8: User preference in the IFOV/OFOV scenarios and slow/fast experiment variants.

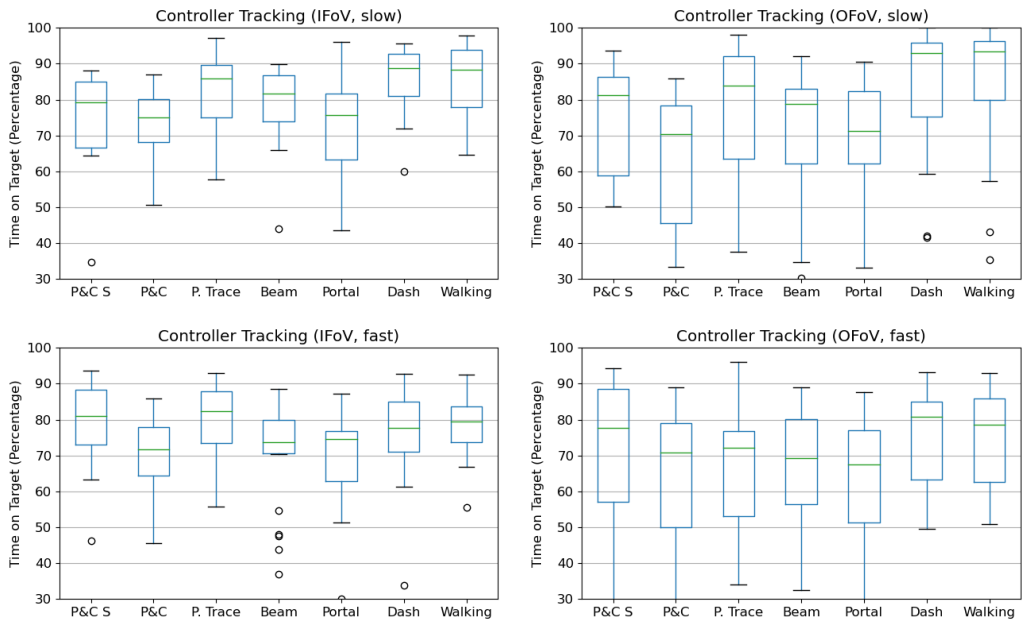


FIGURE B.9: Controller tracking in the IFoV/OFoV scenarios and slow/fast experiment variants.

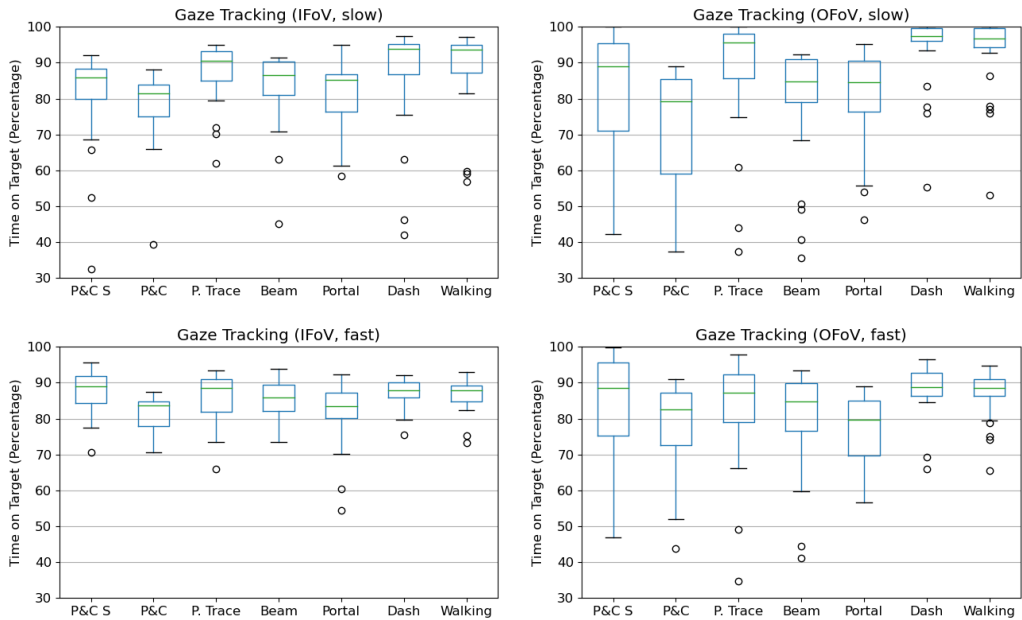


FIGURE B.10: Gaze tracking in the IFoV/OFoV scenarios and slow/fast experiment variants.

# Bibliography

- [1] Haley Adams et al. "Development and evaluation of an immersive virtual reality system for medical imaging of the ear". In: *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*. Vol. 10951. International Society for Optics and Photonics. SPIE, 2019, pp. 265–272.
- [2] Ashu Adhikari et al. "Integrating Continuous and Teleporting VR Locomotion Into a Seamless 'HyperJump' Paradigm". In: *IEEE Transactions on Visualization and Computer Graphics* (2022), pp. 1–17.
- [3] Sarvesh Agrewal et al. "Defining Immersion:: Literature Review and Implications for Research on Audiovisual Experiences". In: *Journal of the Audio Engineering Society* 68.6 (2020), pp. 404–417.
- [4] Majed Al Zayer, Paul MacNeilage, and Eelke Folmer. "Virtual Locomotion: A Survey". In: *IEEE Transactions on Visualization and Computer Graphics* 26.6 (2020), pp. 2315–2334.
- [5] Abdenour Amamra. "GPU-based real-time RGBD data filtering". In: *Journal of Real-Time Image Processing* 14 (Sept. 2014).
- [6] Alba Amato. "Procedural Content Generation in the Game Industry". In: *Game Dynamics*. Springer, 2017, pp. 15–25.
- [7] Steven J. Anbro et al. "Behavioral Assessment in Virtual Reality: An Evaluation of Multi-User Simulations in Healthcare Education". In: *Journal of Organizational Behavior Management* 0.0 (2022), pp. 1–45.
- [8] Víctor H. Andaluz et al. "Multi-user Industrial Training and Education Environment". In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis and Patrick Bourdot. Springer International Publishing, 2018, pp. 533–546.
- [9] David Antón et al. "Augmented Telemedicine Platform for Real-Time Remote Medical Consultation". In: Jan. 2017, pp. 77–89. ISBN: 978-3-319-51810-7.
- [10] Karim Armanious et al. "Adversarial Inpainting of Medical Image Modalities". In: *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3267–3271.
- [11] Knut Augestad and Rolv-Ole Lindsetmo. "Overcoming Distance: Video/- Conferencing as a Clinical and Educational Tool Among Surgeons". In: *World Journal of Surgery* 33 (July 2009), pp. 1356–1365.
- [12] Tycho T de Back et al. "Benefits of immersive collaborative learning in CAVE-based virtual reality". In: *Int. Journal of Educational Technology in Higher Education* 17 (2020), pp. 1–18.
- [13] Gabriel Costa Backes and Tiago Augusto Engel. "Real-Time Massive Terrain Generation using Procedural Erosion on the GPU". In: 2018.

- [14] Paul Balister et al. "River landscapes and optimal channel networks". In: *Proceedings of the National Academy of Sciences* 115.26 (2018), pp. 6548–6553.
- [15] Filippo Bannò et al. "Real-Time Compression of Depth Streams through Meshification and Valence-Based Encoding". In: *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI '12. Singapore, Singapore: Association for Computing Machinery, 2012, 263–270. ISBN: 9781450318259.
- [16] Rosa Baños et al. "Immersion and Emotion: Their Impact on the Sense of Presence". In: *Cyberpsychology & behavior: the impact of the Internet, multimedia and virtual reality on behavior and society* 7 (Jan. 2005), pp. 734–41.
- [17] Woodrow Barfield et al. "Presence and performance within virtual environments". In: *Virtual environments and advanced interface design* (1995), pp. 473–513.
- [18] Andrea Bartl et al. "Affordable But Not Cheap: A Case Study of the Effects of Two 3D-Reconstruction Methods of Virtual Humans". In: *Frontiers in Virtual Reality* 2 (2021).
- [19] Stephan Beck and Bernd Froehlich. "Volumetric calibration and registration of multiple RGBD-sensors into a joint coordinate system". In: *2015 IEEE Symposium on 3D User Interfaces (3DUI)*. 2015, pp. 89–96.
- [20] Stephan Beck et al. "Immersive Group-to-Group Telepresence". In: *IEEE transactions on visualization and computer graphics* 19 (Apr. 2013), pp. 616–25.
- [21] Christopher Beckham and Christopher Pal. "A step towards procedural terrain generation with GANs". In: *arXiv preprint arXiv:1707.03383* (2017).
- [22] Florian Berger et al. "Application of Cinematic Rendering in Clinical Routine CT Examination of Ankle Sprains". In: *American Journal of Roentgenology* 211.4 (Oct. 2018), pp. 887–890.
- [23] N. Berte and Cyril Perrenot. "Surgical apprenticeship in the era of simulation". In: *Journal of Visceral Surgery* 157 (Apr. 2020).
- [24] Jiwan Bhandari, Paul MacNeilage, and Eelke Folmer. "Teleportation without Spatial Disorientation Using Optical Flow Cues". In: *GI '18*. Toronto, Canada: Canadian Human-Computer Communications Society, 2018, 162–167. ISBN: 9780994786838.
- [25] Conrad Fifelski-von Böhlen et al. "Virtual Reality Integrated Multi-Depth-Camera-System for Real-Time Telepresence and Telemanipulation in Caregiving". In: *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. 2020, pp. 294–297.
- [26] Frank Biocca, Chad Harms, and Judee Burgoon. "Towards A More Robust Theory and Measure of Social Presence: Review and Suggested Criteria." In: *Presence* 12 (Oct. 2003), pp. 456–480.
- [27] Gary Bishop and Henry Fuchs. "Research directions in virtual environments: report of an NSF Invitational Workshop, March 23-24, 1992, University of North Carolina at Chapel Hill". In: *COMG*. 1992.
- [28] Roland Blach. "Virtual Reality Technology - An Overview". In: *Product Engineering: Tools and Methods Based on Virtual Reality*. Springer Netherlands, 2008, pp. 21–64.

- [29] Christian Boedecker et al. "Using virtual 3D-models in surgical planning: workflow of an immersive virtual reality application in liver surgery". In: *Langenbeck's archives of surgery* 406 (2021), pp. 911–915.
- [30] Costas Boletsis. "The New Era of Virtual Reality Locomotion: A Systematic Literature Review of Techniques and a Proposed Typology". In: *Multimodal Technologies and Interaction* 1.4 (2017). ISSN: 2414-4088.
- [31] Costas Boletsis and Jarl Cedergren. "VR Locomotion in the New Era of Virtual Reality: An Empirical Comparison of Prevalent Techniques". In: *Advances in Human-Computer Interaction 2019* (Apr. 2019), pp. 1–15.
- [32] Marco Bonali et al. "Surgical Instruments and Preparation of the Specimen". In: *Comparative Atlas of Endoscopic Ear Surgery: Training Techniques Based on an Ovine Model* (2021), pp. 9–27.
- [33] Daniele Bonatto et al. "Explorations for real-time point cloud rendering of natural scenes in virtual reality". In: *2016 International Conference on 3D Imaging (IC3D)*. 2016, pp. 1–7.
- [34] Jean Botev and Steffen Rothkugel. "High-Precision Gestural Input for Immersive Large-Scale Distributed Virtual Environments". In: *Proceedings of the 9th Workshop on Massively Multiuser Virtual Environments*. MMVE'17. Taipei, Taiwan: Association for Computing Machinery, 2017, 7–11.
- [35] Doug A. Bowman, Ryan P. McMahan, and Eric D. Ragan. "Questioning Naturalism in 3D User Interfaces". In: *Commun. ACM* 55.9 (Sept. 2012), 78–88. ISSN: 0001-0782.
- [36] Evren Bozgeyikli et al. "Point & Teleport Locomotion Technique for Virtual Reality". In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. CHI PLAY '16. Austin, Texas, USA: Association for Computing Machinery, 2016, 205–216. ISBN: 9781450344562.
- [37] John Brooke. "SUS: a retrospective". In: *Journal of usability studies* 8.2 (2013), pp. 29–40.
- [38] Rocko Brown and Gregory Pasternack. "How to build a digital river". In: *Earth-Science Reviews* 194 (May 2019).
- [39] Ryan Brucks. *Creating a Volumetric Ray Marcher*. 2016. <https://shaderbits.com/blog/creating-volumetric-ray-marcher>. Accessed: 2020-07-17.
- [40] Antoni Buades, Jose-Luis Lisani, and Marko Miladinović. "Patch-Based Video Denoising With Optical Flow Estimation". In: *IEEE Transactions on Image Processing* 25.6 (2016), pp. 2573–2586.
- [41] Saniye Tugba Bulu. "Place presence, social presence, co-presence, and satisfaction in virtual worlds". In: *Computers & Education* 58.1 (2012), pp. 154–161. ISSN: 0360-1315.
- [42] Fabio Buttussi and Luca Chittaro. "Locomotion in Place in Virtual Reality: A Comparative Evaluation of Joystick, Teleport, and Leaning". In: *IEEE Transactions on Visualization and Computer Graphics* PP (July 2019), pp. 1–1.
- [43] Abraham G. Campbell et al. "Uses of Virtual Reality for Communication in Financial Services: A Case Study on Comparing Different Telepresence Interfaces: Virtual Reality Compared to Video Conferencing". In: *Advances in Information and Communication*. Ed. by Kohei Arai and Rahul Bhatia. Springer International Publishing, 2020, pp. 463–481.

- [44] Chao Cao, Marius Preda, and Titus Zaharia. "3D Point Cloud Compression: A Survey". In: July 2019, pp. 1–9. ISBN: 978-1-4503-6798-1.
- [45] Julie Carmigniani and Borko Furht. "Augmented Reality: An Overview". In: *Handbook of Augmented Reality*. Ed. by Borko Furht. Springer New York, 2011, pp. 3–46.
- [46] Francesco Carrara et al. "Dendritic connectivity controls biodiversity patterns in experimental metacommunities". In: *Proc. of the National Academy of Sciences of the United States of America* 109 (Apr. 2012), pp. 5761–6.
- [47] Luca Carraro et al. "Generation and application of river network analogues for use in ecology and evolution". In: *Ecology and Evolution* 10 (June 2020).
- [48] Polona Caserman, Philipp Achenbach, and Stefan Göbel. "Analysis of Inverse Kinematics Solutions for Full-Body Reconstruction in Virtual Reality". In: *2019 IEEE 7th International Conference on Serious Games and Applications for Health (SeGAH)*. 2019, pp. 1–8.
- [49] Polona Caserman, Augusto Garcia-Agundez, and Stefan Göbel. "A Survey of Full-Body Motion Reconstruction in Immersive Virtual Reality Applications". In: *IEEE Transactions on Visualization and Computer Graphics* 26.10 (2020), pp. 3089–3108.
- [50] Ashley Casey and Victoria A Goodyear. "Can cooperative learning achieve the four learning outcomes of physical education? A review of literature". In: *Quest* 67.1 (2015), pp. 56–72.
- [51] Davide Castelvechi. "Low-cost headsets boost virtual reality's lab appeal". In: *Nature* 533.7602 (2016).
- [52] J. Cecil et al. "Collaborative virtual environments for orthopedic surgery". In: *2013 IEEE International Conference on Automation Science and Engineering (CASE)*. 2013, pp. 133–137.
- [53] Hubert Cecotti. "Cultural Heritage in Fully Immersive Virtual Reality". In: *Virtual Worlds* 1.1 (2022), pp. 82–102. ISSN: 2813-2084.
- [54] Kapil Chalil Madathil and Joel S. Greenstein. "An investigation of the efficacy of collaborative virtual reality systems for moderated remote usability testing". In: *Applied Ergonomics* 65 (2017), pp. 501–514.
- [55] Yadvendar Champawat and Subodh Kumar. "Online point-cloud transmission for tele-immersion". In: Dec. 2012, pp. 79–82.
- [56] Damon Chandler and Sheila Hemami. "VSNR: A wavelet-based Visual Signal-to-Noise Ratio for natural images". In: *Image Processing, IEEE Transactions on* 16 (Oct. 2007), pp. 2284–2298.
- [57] H. Chang et al. "MaskGIT: Masked Generative Image Transformer". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2022, pp. 11305–11315.
- [58] C. Chen, R. Jafari, and N. Kehtarnavaz. "UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor". In: *2015 IEEE International Conference on Image Processing (ICIP)*. Sept. 2015, pp. 168–172.
- [59] H. Chen et al. "Exemplar-based Pattern Synthesis with Implicit Periodic Field Network". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2022, pp. 3698–3707.

- [60] Hanting Chen et al. "Pre-trained image processing transformer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12299–12310.
- [61] Rong Chen et al. "Depth Image Denoising via Collaborative Graph Fourier Transform". In: *Digital TV and Wireless Multimedia Communication*. Ed. by Guangtao Zhai, Jun Zhou, and Xiaokang Yang. Springer Singapore, 2018, pp. 128–137. ISBN: 978-981-10-8108-8.
- [62] Vuthea Chheang et al. "A collaborative virtual reality environment for liver surgery planning". In: *Computers & Graphics* 99 (2021), pp. 234–246.
- [63] Vuthea Chheang et al. "Collaborative Virtual Reality for Laparoscopic Liver Surgery Training". In: *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. 2019, pp. 1–17.
- [64] Vuthea Chheang et al. "Group WiM: A Group Navigation Technique for Collaborative Virtual Reality Environments". In: *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2022, pp. 556–557.
- [65] SungIk Cho et al. "Effects of volumetric capture avatars on social presence in immersive virtual environments". In: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2020, pp. 26–34.
- [66] Nicklas Christensen et al. "Feasibility of Team Training in Virtual Reality for Robot-Assisted Minimally Invasive Surgery". In: Apr. 2018, pp. 1–4.
- [67] Chris G. Christou and Poppy Aristidou. "Steering Versus Teleport Locomotion for Head Mounted Displays". In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Springer International Publishing, 2017, pp. 431–446.
- [68] Pietro Cipresso et al. "The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature". In: *Frontiers in Psychology* 9 (Nov. 2018), p. 2086.
- [69] Arnis Cirulis, Lauris Taube, and Zintis Erics. "Automated Generation of Digital Twin in Virtual Reality for Interaction with Specific Nature Ecosystem". In: *Universal Access in Human-Computer Interaction. User and Context Diversity*. Ed. by Margherita Antona and Constantine Stephanidis. Springer International Publishing, 2022, pp. 187–202.
- [70] Jeremy Clifton and Stephen Palmisano. "Effects of Steering Locomotion and Teleporting on Cybersickness and Presence in HMD-Based Virtual Reality". In: *Virtual Real.* 24.3 (Sept. 2020), 453–468. ISSN: 1359-4338.
- [71] Sebastian Cmentowski, Andrey Krekhov, and Jens Krüger. "Outstanding: A Multi-Perspective Travel Approach for Virtual Reality Games". In: *Proc. of the Annual Symposium on Computer-Human Interaction in Play. CHI PLAY '19*. Barcelona, Spain: Association for Computing Machinery, 2019, 287–299. ISBN: 9781450366885.
- [72] Anthony M Codd and Bipasha Choudhury. "Virtual reality anatomy: Is it comparable with traditional methods in the teaching of human forearm musculoskeletal anatomy?" In: *Anatomical sciences education* 4.3 (2011), pp. 119–125.

- [73] Charlotte Coles. *Virtual Reality: The Most Disruptive Technology of the Next Decade*. Research article on IDTechEx. 2020. <https://www.idtechex.com/it/research-article/virtual-reality-the-most-disruptive-technology-of-the-next-decade/22183>. Last accessed on 25. January 2023.
- [74] Yann Collet. *Zstandard - a fast real-time compression algorithm*. Github repository. 2016. <https://github.com/facebook/zstd>; accessed on 26. Februar 2020.
- [75] Guillaume Cordonnier et al. "Large Scale Terrain Generation from Tectonic Uplift and Fluvial Erosion". In: *Computer Graphics Forum* 35 (May 2016).
- [76] Yann Cortial et al. "Procedural Tectonic Planets". In: *Computer Graphics Forum* 38 (May 2019), pp. 1–11.
- [77] Evelyn Dappa et al. "Cinematic rendering – an alternative to volume rendering for 3D computed tomography imaging". In: *Insights into Imaging* 7 (Sept. 2016).
- [78] Tycho T De Back, Angelica M Tinga, and Max M Louwse. "Learning in immersed collaborative virtual environments: design and implementation". In: *Interactive Learning Environments* (2021), pp. 1–19.
- [79] Aikaterini Dedeilia et al. "Medical and Surgical Education Challenges and Innovations in the COVID-19 Era: A Systematic Review". In: *In Vivo* 34 (June 2020), pp. 1603–1611.
- [80] Teng Deng et al. "Multiple consumer-grade depth camera registration using everyday objects". In: *Image and Vision Computing* 62 (2017), pp. 1–7.
- [81] Ye Deng et al. "T-Former: An Efficient Transformer for Image Inpainting". In: *Proceedings of the 30th ACM International Conference on Multimedia*. MM '22. Association for Computing Machinery, 2022, pp. 6559–6568.
- [82] Evgenij Derzapf et al. "River Networks for Instant Procedural Planets". In: *Computer Graphics Forum* 30 (Nov. 2011), pp. 2031–2040.
- [83] Oliver Deussen et al. "Realistic Modeling and Rendering of Plant Ecosystems". In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM. 1998, pp. 275–286.
- [84] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.
- [85] Massimiliano Di Luca et al. "Locomotion Vault: The Extra Mile in Analyzing VR Locomotion Techniques". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Association for Computing Machinery, 2021.
- [86] Cambridge Dictionary. *Definition of confusing*. <https://dictionary.cambridge.org/de/worterbuch/englisch/confusing>. Accessed: 2023-03-29.
- [87] Collins Dictionary. *Definition of confusing*. <https://www.collinsdictionary.com/de/worterbuch/englisch/confusing>. Accessed: 2023-03-29.
- [88] Sylvie Dijkstra-Soudarissanane et al. "Multi-Sensor Capture and Network Processing for Virtual Reality Conferencing". In: *Proceedings of the 10th ACM Multimedia Systems Conference*. MMSys '19. Amherst, Massachusetts: Association for Computing Machinery, 2019, 316–319.



- [89] Guanting Dong, Yueyi Zhang, and Zhiwei Xiong. "Spatial Hierarchy Aware Residual Pyramid Network for Time-of-Flight Depth Denoising". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Springer International Publishing, 2020, pp. 35–50. ISBN: 978-3-030-58586-0.
- [90] Mingsong Dou et al. "Fusion4D: Real-time Performance Capture of Challenging Scenes". In: *ACM Transactions on Graphics* 35 (July 2016).
- [91] Mingsong Dou et al. "Motion2fusion: Real-Time Volumetric Performance Capture". In: *ACM Trans. Graph.* 36.6 (Nov. 2017).
- [92] Antoine Dricot and João Ascenso. "Hybrid Octree-Plane Point Cloud Geometry Coding". In: *27th European Signal Processing Conference (EUSIPCO)*. 2019, pp. 1–5.
- [93] Antoine Dricot, Fernando Pereira, and João Ascenso. "Rate-Distortion Driven Adaptive Partitioning for Octree-Based Point Cloud Geometry Coding". In: *25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 2969–2973.
- [94] Alexander Duda and Udo Frese. "Accurate Detection and Localization of Checkerboard Corners for Calibration". In: *British Machine Vision Conference*. Sept. 2018.
- [95] Gilles Dussault and Maria Franceschini. "Not enough there, too many here: Understanding geographical imbalances in the distribution of the health workforce". In: *Human resources for health* 4 (Feb. 2006), p. 12.
- [96] Lars C. Ebert et al. "Forensic 3D Visualization of CT Data Using Cinematic Volume Rendering: A Preliminary Study". In: *American Journal of Roentgenology* 208.2 (Feb. 2017), pp. 233–240.
- [97] Clemens Eisank, Mike Smith, and John Hillier. "Assessment of multiresolution segmentation for delimiting drumlins in digital elevation models". In: *Geomorphology* (June 2014).
- [98] Somayya Elmoghazy et al. "Survey of Immersive Techniques for Surgical Care Telemedicine Applications". In: June 2021, pp. 1–6.
- [99] Carmine Elvezio et al. "Collaborative Virtual Reality for Low-Latency Interaction". In: *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*. Oct. 2018, pp. 179–181.
- [100] David Englmeier, Wanja Sajko, and Andreas Butz. "Spherical World in Miniature: Exploring the Tiny Planets Metaphor for Discrete Locomotion in Virtual Reality". In: *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 2021, pp. 345–352.
- [101] Mohamed Estai and Stuart Bunt. "Best teaching practices in anatomy education: A critical review". In: *Annals of Anatomy-Anatomischer Anzeiger* 208 (2016), pp. 151–157.
- [102] Chris Evans. *IL Gigante: Michelangelo's David in VR*. (Talk) VR experience that digitally reproduces Michelangelo's David. 2018. <https://www.gdcvault.com/play/1025161/contactUs>. Last accessed 05 June 2023.
- [103] Chris Evans et al. *IL DIVINO: Michelangelo's Sistine Ceiling in VR*. VR experience that digitally reproduces the Sistine Chapel. 2019. <http://www.sistinevr.com/>. Last accessed 05 June 2023.

- [104] Experius VR. *Nefertari: Journey to Eternity*. (Article) VR experience that digitally reproduces Nefertari's tomb in 3D. 2018. <https://timeandhistory.com/nefertari-journey-to-eternity/>. Last accessed 05 June 2023.
- [105] Marta Fairén González et al. "Virtual reality to teach anatomy". In: *Eurographics 2017: education papers*. European Association for Computer Graphics (Eurographics). 2017, pp. 51–58.
- [106] Balázs Faludi et al. "Direct Visual and Haptic Volume Rendering of Medical Data Sets for an Immersive Exploration in Virtual Reality". In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*. Springer International Publishing, 2019, pp. 29–37.
- [107] Xuan Fei et al. "Perceptual image quality assessment based on structural similarity and visual masking". In: *Signal Processing: Image Communication* 27.7 (2012), pp. 772–783.
- [108] Yu Feng, Shaoshan Liu, and Yuhao Zhu. "Real-Time Spatio-Temporal LiDAR Point Cloud Compression". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10766–10773.
- [109] Elisabeth M. Feudner et al. "Virtual reality training improves wet-lab performance of capsulorhexis: results of a randomized, controlled study". In: *Graefe's Archive for Clinical and Experimental Ophthalmology* 247.7 (Jan. 2009), p. 955.
- [110] Elliot K. Fishman et al. "Volume Rendering versus Maximum Intensity Projection in CT Angiography: What Works Best, When, and Why". In: *RadioGraphics* 26.3 (2006), pp. 905–922.
- [111] Gerd Flodgren et al. "Interactive telemedicine: effects on professional practice and health care outcomes". In: *The Cochrane database of systematic reviews* 9 (Sept. 2015), p. CD002098.
- [112] Jann Philipp Freiwald et al. "Effects of Avatar Appearance and Locomotion on Co-Presence in Virtual Reality Collaborations". In: *Proceedings of Mensch Und Computer 2021*. MuC '21. Ingolstadt, Germany: Association for Computing Machinery, 2021, 393–401. ISBN: 9781450386456.
- [113] Jann Philipp Freiwald et al. "The Continuity of Locomotion: Rethinking Conventions for Locomotion and Its Visualization in Shared Virtual Reality Spaces". In: *ACM Trans. Graph.* 41.6 (Nov. 2022). ISSN: 0730-0301.
- [114] F. A. Fridriksson et al. "Become your Avatar: Fast Skeletal Reconstruction from Sparse Data for Fully-tracked VR". In: *ICAT-EGVE 2016 - Posters and Demos*. The Eurographics Association, 2016.
- [115] Ryoske Fujii, Ryo Hachiuma, and Hideo Saito. "RGB-D Image Inpainting Using Generative Adversarial Network with a Late Fusion Approach". In: *ArXiv abs/2110.07413* (2020).
- [116] James Gain, Patrick Marais, and Wolfgang Straßer. "Terrain sketching". In: Jan. 2009, pp. 31–38.
- [117] James Gain, B. Merry, and Patrick Marais. "Parallel, Realistic and Controllable Terrain Synthesis". In: *Computer Graphics Forum* 34 (May 2015).
- [118] Bruno Galerne et al. "Gabor Noise by Example". In: *ACM Trans. Graph.* 31.4 (July 2012).
- [119] Eric Galin et al. "A Review of Digital Terrain Modeling". In: *Computer Graphics Forum* (2019).

- [120] Dominik Gall et al. "Embodiment in Virtual Reality Intensifies Emotional Responses to Virtual Stimuli". In: *Frontiers in Psychology* 12 (Sept. 2021).
- [121] Luigi Gallo et al. "Comparative evaluation of methods for filtering Kinect depth data". In: *Multimedia Tools and Applications* 74 (May 2014).
- [122] Guillaume Gamelin et al. "Point-cloud avatars to improve spatial communication in immersive collaborative virtual environments". In: *Personal and Ubiquitous Computing* 25 (June 2021).
- [123] Manuel N Gamito and Steve C Maddock. "Accurate Multi-Dimensional Poisson-Disk Sampling". In: *ACM Transactions on Graphics (TOG)* 29.1 (2009), 8:1–8:19.
- [124] Danilo Gasques et al. "ARTEMIS: A Collaborative Mixed-Reality System for Immersive Surgical Telementoring". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966.
- [125] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2414–2423.
- [126] Daphne Geerse, Bert Coolen, and Melvyn Roerdink. "Kinematic Validation of a Multi-Kinect v2 Instrumented 10-Meter Walkway for Quantitative Gait Assessments". In: *PLOS ONE* 10 (Oct. 2015), e0139913.
- [127] Jean-David Génevaux et al. "Terrain generation using procedural models based on hydrology". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 143.
- [128] Fabian Giesen. *Implementation of several rANS variants*. Repository on Github. 2014. [https://github.com/rygorous/ryg\\_rans](https://github.com/rygorous/ryg_rans); Accessed on 26. Februar 2020.
- [129] Guillaume Gilet et al. "Local Random-Phase Noise for Procedural Texturing". In: *ACM Trans. Graph.* 33.6 (Nov. 2014).
- [130] Philip A. Glemser et al. "A New Approach for Photorealistic Visualization of Rendered Computed Tomography Images". In: *World Neurosurgery* 114 (2018), e283–e292. ISSN: 1878-8750.
- [131] Kilian Gloy et al. "Immersive anatomy atlas: Learning factual medical knowledge in a virtual reality environment". In: *Anatomical Sciences Education* 15.2 (2022), pp. 360–368.
- [132] Google Arts and Culture, makemepulse. *VersaillesVR: The Palace Is Yours*. VR experience that digitally reproduces the Versailles palace in 3D. 2019. <https://artsandculture.google.com/project/versailles>. Accessed: 05 June 2023.
- [133] Nathan Navarro Griffin and Eelke Folmer. "Out-of-Body Locomotion: Vectionless Navigation with a Continuous Avatar Representation". In: *25th ACM Symposium on Virtual Reality Software and Technology*. VRST '19. Parramatta, NSW, Australia: Association for Computing Machinery, 2019.
- [134] Guest Editor: Gernot Groemer. "Simulating Mars on Earth". In: *Astrobiology* 14.5 (2014), pp. 357–359.
- [135] Éric Guérin et al. "Interactive Example-based Terrain Authoring with Conditional Generative Adversarial Networks". In: *ACM Trans. Graph.* 36.6 (2017), 228:1–228:13.

- [136] Simon N. B. Gunkel et al. "From 2D to 3D video conferencing: modular RGB-D capture and reconstruction for interactive natural user representations in immersive extended reality (XR) communication". In: *Frontiers in Signal Processing* 3 (2023). ISSN: 2673-8198.
- [137] Kaiwen Guo et al. "Real-Time Geometry, Albedo, and Motion Reconstruction Using a Single RGB-D Camera". In: *ACM Trans. Graph.* 36.4 (July 2017).
- [138] Yong Guo et al. "CT two-dimensional reformation versus three-dimensional volume rendering with regard to surgical findings in the preoperative assessment of the ossicular chain in chronic suppurative otitis media". In: *European journal of radiology* 82.9 (Sept. 2013), 1519—1524.
- [139] Manoj Gupta, Jörg Lechner, and Basant Agarwal. "Performance Analysis of Kalman Filter in Computed Tomography Thorax for Image Denoising". In: *Recent Patents on Computer Science* 12 (2019), pp. 1–1.
- [140] H. Hamout and A. Elyousfi. "Fast Depth Map Intra Coding for 3D Video Compression Based Tensor Feature Extraction and Data Analysis". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2019), pp. 1–1. ISSN: 1558-2205.
- [141] Vincent Havard et al. "Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations". In: *Production & Manufacturing Research* 7.1 (2019), pp. 472–489.
- [142] Eric Heitz and Fabrice Neyret. "High-Performance By-Example Noise Using a Histogram-Preserving Blending Operator". In: *Proc. ACM Comput. Graph. Interact. Tech.* 1.2 (Aug. 2018).
- [143] S. Hemanth Kumar and K. R. Ramakrishnan. "Depth compression via planar segmentation". In: *Multimedia Tools and Applications* 78 (July 2018).
- [144] Mark Hendrikx et al. "Procedural content generation for games: A survey". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9.1 (2013), p. 1.
- [145] F. Hernell, P. Ljung, and A. Ynnerman. "Local Ambient Occlusion in Direct Volume Rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 16.4 (2010), pp. 548–559.
- [146] Jonathan Ho et al. "Cascaded diffusion models for high fidelity image generation". In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 2249–2281.
- [147] Dirk Holz and Sven Behnke. "Fast Range Image Segmentation and Smoothing Using Approximate Surface Reconstruction and Region Growing". In: vol. 194. June 2012. ISBN: 978-3-642-33931-8.
- [148] Patrik Huber et al. "A multiresolution 3d morphable face model and fitting framework". In: *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. University of Surrey. 2016.
- [149] Tuomo Hyttinen. "Terrain Synthesis Using Noise". MA thesis. University of Tampere, 2017.
- [150] IDC. *Worldwide Augmented and Virtual Reality Spending Guide*. 2022. <https://www.idc.com/getdoc.jsp?containerId=prUS49916122>; Accessed on 25. January 2023.

- [151] Markus Ihmsen et al. "SPH Fluids in Computer Graphics". In: *Eurographics 2014 - State of the Art Reports*. Ed. by Sylvain Lefebvre and Michela Spagnuolo. The Eurographics Association, 2014.
- [152] Wijnand Ijsselstein and Giuseppe Riva. "Being There: The Experience of Presence in Mediated Environments". In: *EMERGING COMMUNICATION 5* (Jan. 2003), pp. 3–16.
- [153] ImmersiveTouch Inc. *ImmersiveView: (Screencapture) feature demonstration video of the VR medical visualization software*. 2018. <https://www.immersivetouch.com/immersiveview-vr>. Last accessed on 25. January 2023.
- [154] Fortune Business Insights. *Market Research Report: Virtual Reality Market Size, Share & covid-19 Impact Analysis*. 2022. <https://www.fortunebusinessinsights.com/industry-reports/virtual-reality-market-101378>. Accessed: 25. January 2023.
- [155] Ayesha Irfan, Adeel Zafar, and Shahbaz Hassan. "Evolving Levels for General Games Using Deep Convolutional Generative Adversarial Networks". In: *2019 11th Computer Science and Electronic Engineering (CEECE)*. 2019, pp. 96–101.
- [156] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 5967–5976.
- [157] Balázs Jákó. "Fast Hydraulic and Thermal Erosion on GPU". In: *Eurographics 2011 - Short Papers*. 2011.
- [158] Euee S. Jang et al. "Video-Based Point-Cloud-Compression Standard in MPEG: From Evidence Collection to Committee Draft [Standards in a Nutshell]". In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 118–123.
- [159] Julian Jang-Jaccard et al. "WebRTC-based video conferencing service for telehealth". In: *Computing* 98 (Jan. 2016), pp. 169–193.
- [160] Jinwoo Jeon et al. "Struct-MDC: Mesh-refined unsupervised depth completion leveraging structural regularities from visual SLAM". In: *IEEE Robot. and Autom. Letters* 7.3 (2022), pp. 6391–6398.
- [161] Boyi Jiang et al. "SelfRecon: Self Reconstruction Your Digital Avatar From Monocular Video". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 5605–5615.
- [162] Fan Jiang, Xubo Yang, and Lele Feng. "Real-Time Full-Body Motion Reconstruction and Recognition for off-the-Shelf VR Devices". In: *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1. VRCAI '16*. Zhuhai, China: Association for Computing Machinery, 2016, 309–318.
- [163] Wu Jin, Li Zun, and Liu Yong. "Double-Constraint Inpainting Model of a Single-Depth Image". In: *Sensors* 20.6 (2020).
- [164] Michal Joachimczak, Juan Liu, and Hiroshi Ando. "Real-Time Mixed-Reality Telepresence via 3D Reconstruction with HoloLens and Commodity Depth Sensors". In: *Proceedings of the 19th ACM International Conference on Multimodal Interaction. ICMI '17*. Glasgow, UK: Association for Computing Machinery, 2017, 514–515. ISBN: 9781450355438.
- [165] David W Johnson, Roger T Johnson, and Mary Beth Stanne. "Cooperative learning methods: A meta-analysis". In: (2000).

- [166] Brett Jones et al. "RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-Camera Units". In: Oct. 2014.
- [167] Thouis R Jones. "Efficient Generation of Poisson-Disk Sampling Patterns". In: *Journal of Graphics Tools* 11.2 (2006), pp. 27–36.
- [168] Justin Joseph and R. Periyasamy. "An image driven bilateral filter with adaptive range and spatial parameters for denoising Magnetic Resonance Images". In: *Computers & Electrical Engineering* 69 (2018), pp. 782–795. ISSN: 0045-7906.
- [169] Hanseul Jun and Jeremy Bailenson. "Temporal RVL: A Depth Stream Compression Method". In: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2020, pp. 664–665.
- [170] Younhyun Jung et al. "A direct volume rendering visualization approach for serial PET–CT scans that preserves anatomical consistency". In: *International Journal of Computer Assisted Radiology and Surgery* 14.5 (May 2019), pp. 733–744.
- [171] Michael Kalbermatten et al. "Multiscale analysis of geomorphological and geological features in high resolution digital elevation models using the wavelet transform". In: *Geomorphology* 138.1 (2012), pp. 352–363.
- [172] Maximilian Kaluschke et al. "A Shared Haptic Virtual Environment for Dental Surgical Skill Training". In: *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)* (2021), pp. 347–352.
- [173] Maximilian Kaluschke et al. "HIPS – A Virtual Reality Hip Prosthesis Implantation Simulator". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. Mar. 2018.
- [174] Ahmed Kamal and Carlos Andujar. "Designing, testing and adapting navigation techniques for the immersive web". In: *Computers & Graphics* 106 (2022), pp. 66–76. ISSN: 0097-8493.
- [175] Keito Kamimura et al. "Teleclinical Support System via MR-HMD Displaying Doctor's Instructions and Patient Information". In: Mar. 2021, pp. 477–479.
- [176] Julius Kammerl et al. "Real-time compression of point cloud streams". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 778–785.
- [177] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: June 2020, pp. 8107–8116.
- [178] Shunichi Kasahara et al. "Malleable Embodiment: Changing Sense of Embodiment by Spatial-Temporal Deformation of Virtual Human Body". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: Association for Computing Machinery, 2017, 6438–6448.
- [179] Sujay Kawale and John Ferris. "Developing a Methodology to Synthesize Terrain Profiles and Evaluate their Statistical Properties". In: *SAE Technical Papers* (Apr. 2011).
- [180] Alex Kelley, Michael Malin, and Gregory Nielson. "Terrain simulation using a model of stream erosion". In: vol. 22. Aug. 1988, pp. 263–268.
- [181] Jonathan W. Kelly et al. "Teleporting through virtual environments: Effects of path scale and environment scale on spatial updating". In: *IEEE Transactions on Visualization and Computer Graphics* 26.5 (2020), pp. 1841–1850.

- [182] Thomas P. Kersten et al. "Virtual Reality for Cultural Heritage Monuments – from 3D Data Recording to Immersive Visualisation". In: *Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection*. Ed. by Marinos Ioannides et al. Springer International Publishing, 2018, pp. 74–83.
- [183] Os Keyes et al. "Reimagining (women's) health: HCI, gender and essentialised embodiment". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 27.4 (2020), pp. 1–42.
- [184] Ahmed Khalifa et al. "PCGRL: Procedural Content Generation via Reinforcement Learning". In: *Artificial Intelligence and Interactive Digital Entertainment Conference*. 2020.
- [185] Konstantina Kilteni, Raphaela Groten, and Mel Slater. "The Sense of Embodiment in Virtual Reality". In: *Presence Teleoperators & Virtual Environments* 21 (Nov. 2012).
- [186] Hansung Kim et al. "Outdoor Dynamic 3-D Scene Reconstruction". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.11 (2012), pp. 1611–1622.
- [187] Naimin Koh, Pradeep Kumar Jayaraman, and Jianmin Zheng. "Parallel Point Cloud Compression Using Truncated Octree". In: *2020 International Conference on Cyberworlds (CW)*. 2020, pp. 1–8.
- [188] Jan Kolkmeier et al. "With a little help from a holographic friend: the Open-IMPRESS mixed reality telepresence toolkit for remote collaboration systems". In: Nov. 2018, pp. 1–11.
- [189] Igor Kozak et al. "Virtual reality simulator for vitreoretinal surgery using integrated OCT data". In: *Clinical ophthalmology (Auckland, N.Z.)* 8 (Mar. 2014), pp. 669–672.
- [190] Arno Krüeger et al. "Sinus Endoscopy - Application of Advanced GPU Volume Rendering for Virtual Endoscopy". In: *IEEE transactions on visualization and computer graphics* 14 (Jan. 2009), pp. 1491–8.
- [191] Lucie Kruse et al. "On the Use of Jumping Gestures for Immersive Teleportation in VR". In: Dec. 2020.
- [192] Ramazan Kurul et al. "An alternative method for anatomy training: Immersive virtual reality". In: *Anatomical Sciences Education* 13.5 (2020), pp. 648–656.
- [193] Joseph Kvedar, Molly Coye, and Wendy Everett. "Connected Health: A Review Of Technologies And Strategies To Improve Patient Care With Telemedicine And Telehealth". In: *Health affairs (Project Hope)* 33 (Feb. 2014), pp. 194–9.
- [194] Eva Kyndt et al. "A meta-analysis of the effects of face-to-face cooperative learning. Do recent studies falsify or verify earlier findings?" In: *Educational research review* 10 (2013), pp. 133–149.
- [195] Marjan Laal and Seyed Mohammad Ghodsi. "Benefits of collaborative learning". In: *Procedia - Social and Behavioral Sciences* 31 (2012). World Conference on Learning, Teaching & Administration - 2011, pp. 486–490.
- [196] Marjan Laal and Seyed Mohammad Ghodsi. "Benefits of collaborative learning". In: *Procedia-social and behavioral sciences* 31 (2012), pp. 486–490.
- [197] Ares Lagae et al. "A Survey of Procedural Noise Functions". In: *Computer Graphics Forum* 29 (Dec. 2010).

- [198] Chengyuan Lai et al. "The Cognitive Loads and Usability of Target-based and Steering-based Travel Techniques". In: *IEEE Transactions on Visualization and Computer Graphics* PP (Aug. 2021), pp. 1–1.
- [199] Brendan Lane, Przemyslaw Prusinkiewicz, et al. "Generating Spatial Distributions for Multilevel Models of Plant Communities". In: *Graphics Interface 2002 Conference*. Citeseer. 2002, pp. 69–80.
- [200] Christian Larsen et al. "The efficacy of virtual reality simulation training in laparoscopy: A systemic review of randomized trials". In: *Acta obstetrica et gynecologica Scandinavica* 91 (June 2012), pp. 1015–28.
- [201] Abdul Latif et al. "A Critical Evaluation of Procedural Content Generation Approaches for Digital Twins". In: *J. Sensors* 2022 (2022), pp. 1–13.
- [202] Rifat Latifi et al. "Telemedicine and telepresence for trauma and emergency management". In: *Scandinavian journal of surgery : SJS : official organ for the Finnish Surgical Society and the Scandinavian Surgical Society* 96 (Feb. 2007), pp. 281–9.
- [203] Marc Erich Latoschik et al. "The Effect of Avatar Realism in Immersive Social Virtual Realities". In: *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. Association for Computing Machinery, 2017. ISBN: 9781450355483.
- [204] Joseph J. LaViola. "A Discussion of Cybersickness in Virtual Environments". In: *SIGCHI Bull.* 32.1 (Jan. 2000), 47–56. ISSN: 0736-6906.
- [205] Ji Eun Lee et al. "Effectiveness of virtual reality-based learning for anatomy education: A systematic review and meta-analysis". In: *Anatomical Sciences Education* 14.5 (2021), pp. 587–600.
- [206] Juyoung Lee, Sang Chul Ahn, and Jae-In Hwang. "A Walking-in-Place Method for Virtual Reality Using Position and Orientation Tracking". In: *Sensors* 18.9 (2018).
- [207] Sihaeng Lee et al. "Multi-Scaled and Densely Connected Locally Convolutional Layers for Depth Completion". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 8360–8367.
- [208] Paolo Leoncini et al. "Multiple NUI Device Approach to Full Body Tracking for Collaborative Virtual Environments". In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis, Patrick Bourdot, and Antonio Mongelli. Springer International Publishing, 2017, pp. 131–147.
- [209] Dominic Lesaca et al. "Comparing Teleportation Methods for Travel in Everyday Virtual Reality". In: *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2022, pp. 238–242.
- [210] Jie Li et al. "Designing a Social VR Clinic for Medical Consultations". In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI EA '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, 1–9.
- [211] Peng Li et al. "Evaluation of the ICP Algorithm in 3D Point Cloud Registration". In: *IEEE Access* 8 (2020), pp. 68030–68048.
- [212] Sijin Li et al. "Generating Terrain Data for Geomorphological Analysis by Integrating Topographical Features and Conditional Generative Adversarial Networks". In: *Remote Sensing* 14 (Feb. 2022).



- [213] Wenbo Li et al. "MAT: Mask-Aware Transformer for Large Hole Image Inpainting". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10748–10758.
- [214] Xi-zhi Li, René Weller, and Gabriel Zachmann. "Procedural 3D Asteroid Surface Detail Synthesis". In: *Eurographics 2020 - Short Papers*. Ed. by Alexander Wilkie and Francesco Banterle. The Eurographics Association, 2020. ISBN: 978-3-03868-101-4.
- [215] Xi-zhi Li, René Weller, and Gabriel Zachmann. "AstroGen—Procedural Generation of Highly Detailed Asteroid Models". In: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1771–1778.
- [216] Yang LI et al. "Gesture interaction in virtual reality". In: *Virtual Reality & Intelligent Hardware* 1.1 (2019), pp. 84–112.
- [217] Ziheng Li et al. "Promising Generative Adversarial Network Based Sinogram Inpainting Method for Ultra-Limited-Angle Computed Tomography Imaging". In: *Sensors* 19.18 (2019).
- [218] Zun Li and Jin Wu. "Learning Deep CNN Denoiser Priors for Depth Image Inpainting". In: *Applied Sciences* 9.6 (2019).
- [219] Antonios Liapis et al. "Transforming exploratory creativity with DeLeNoX". In: *Proceedings of the 4th International Conference on Computational Creativity, ICC3 2013*. Ed. by Mary Lou Maher et al. Proceedings of the 4th International Conference on Computational Creativity, ICC3 2013. Faculty of Architecture, Design and Planning, The University of Sydney, 2013, pp. 56–63.
- [220] Bor-Shing Lin et al. "Temporal and Spatial Denoising of Depth Maps". In: *Sensors (Basel, Switzerland)* 15 (Aug. 2015), pp. 18506–25.
- [221] Jenny Lin et al. "A virtual reality platform for dynamic human-scene interaction". In: Nov. 2016, pp. 1–4.
- [222] Guilin Liu. *PyTorch Implementation of the Partial Convolution Layer for Padding and Image Inpainting*. 2018. <https://github.com/NVIDIA/partialconv>, Accessed on 23. July 2023.
- [223] Guilin Liu et al. "Image Inpainting for Irregular Holes Using Partial Convolutions". In: *European Conference on Computer Vision*. 2018.
- [224] Jun Liu et al. "Perception-driven procedural texture generation from examples". In: *Neurocomputing* 291 (2018), pp. 21–34.
- [225] Junshan Liu et al. "Static Terrestrial Laser Scanning (TLS) for Heritage Building Information Modeling (HBIM): A Systematic Review". In: *Virtual Worlds* 2.2 (2023), pp. 90–114. ISSN: 2813-2084.
- [226] Yunpeng Liu et al. "Hybrid Lossless-Lossy Compression for Real-Time Depth-Sensor Streams in 3D Telepresence Applications". In: *Advances in Multimedia Information Processing – PCM 2015*. Springer International Publishing, 2015, pp. 442–452. ISBN: 978-3-319-24075-6.
- [227] Matthew Lombard and Theresa Ditton. "At the Heart of It All: The Concept of Presence". In: *J. Comput. Mediat. Commun.* 3 (2006), p. 0.
- [228] Stefan Maas et al. "A mixed reality telemedicine system for collaborative ultrasound diagnostics and ultrasound-guided interventions". In: *AboutOpen* 9 (Apr. 2022), pp. 15–20.

- [229] M. Magdics, D. White, and S. Marks. "Extending a Virtual Reality Nasal Cavity Education Tool with Volume Rendering". In: *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. 2018, pp. 811–814.
- [230] Ilya Makarov and Gleb Borisenko. "Depth Inpainting via Vision Transformer". In: *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. Oct. 2021, pp. 286–291.
- [231] Guido Makransky, Lau Lilleholt, and Anders Aaby. "Development and validation of the Multimodal Presence Scale for virtual reality environments: A confirmatory factor analysis and item response theory approach". In: *Computers in Human Behavior* 72 (2017), pp. 276–285. ISSN: 0747-5632.
- [232] Fangchang Mal and Sertac Karaman. "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1–8.
- [233] Peter M. Maloca et al. "High-Performance Virtual Reality Volume Rendering of Original Optical Coherence Tomography Point-Cloud Data Enhanced With Real-Time Ray Casting". In: *Translational vision science & technology* 7.4 (July 2018), pp. 2–2.
- [234] Jacob C. Mandell et al. "Clinical Applications of a CT Window Blending Algorithm: RADIO (Relative Attenuation-Dependent Image Overlay)". In: *Journal of Digital Imaging* 30.3 (June 2017), pp. 358–368.
- [235] Aihua Mao et al. "Easy and Fast Reconstruction of a 3D Avatar with an RGB-D Sensor". In: *Sensors (Switzerland)* 17 (May 2017).
- [236] Douglas B Markant et al. "Enhanced memory as a common effect of active learning". In: *Mind, Brain, and Education* 10.3 (2016), pp. 142–152.
- [237] Esteban Segarra Martinez, Annie S. Wu, and Ryan P. McMahan. "Research Trends in Virtual Reality Locomotion Techniques". In: *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2022, pp. 270–280.
- [238] Mohamed Marzouk. "Using 3D Laser Scanning to Analyze Heritage Structures: The Case Study of Egyptian Palace". In: *Journal of Civil Engineering and Management* 26 (Jan. 2020), pp. 53–65.
- [239] Elias Matsas, George-C. Vosniakos, and Dimitrios Batras. "Modelling Simple Human-Robot Collaborative Manufacturing Tasks in Interactive Virtual Environments". In: *Proceedings of the 2016 Virtual Reality International Conference. VRIC '16*. Laval, France: Association for Computing Machinery, 2016.
- [240] Keigo Matsumoto et al. "Unlimited Corridor: Redirected Walking Techniques Using Visuo Haptic Interaction". In: *ACM SIGGRAPH 2016 Emerging Technologies. SIGGRAPH '16*. Anaheim, California: Association for Computing Machinery, 2016. ISBN: 9781450343725.
- [241] Jesus Mayor, Laura Raya, and Alberto Sanchez. "A Comparative Study of Virtual Reality Methods of Interaction and Locomotion Based on Presence, Cybersickness, and Usability". In: *IEEE Transactions on Emerging Topics in Computing* 9.3 (2021), pp. 1542–1553.
- [242] Jillian McGrath et al. "Using Virtual Reality Simulation Environments to Assess Competence for Emergency Medicine Learners". In: *Academic Emergency Medicine* 25 (Sept. 2017).

- [243] S. Mehrotra et al. "Low-complexity, near-lossless coding of depth maps from kinect-like depth cameras". In: *2011 IEEE 13th International Workshop on Multimedia Signal Processing*. Oct. 2011, pp. 1–6.
- [244] Xing Mei, Philippe Decaudin, and Bao-Gang Hu. "Fast Hydraulic Erosion Simulation and Visualization on GPU". In: Oct. 2007, pp. 47–56. ISBN: 978-0-7695-3009-3.
- [245] R. Mekuria, K. Blom, and P. Cesar. "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.4 (Apr. 2017), pp. 828–842. ISSN: 1558-2205.
- [246] Rufael Mekuria, Kees Blom, and Puente César. "Design, Implementation and Evaluation of a Point Cloud Codec for Tele-Immersive Video". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27 (Jan. 2016), pp. 1–1.
- [247] Rufael Mekuria et al. "A 3D Tele-Immersion System Based on Live Captured Mesh Geometry". In: *Proceedings of the 4th ACM Multimedia Systems Conference*. MMSys '13. Oslo, Norway: Association for Computing Machinery, 2013, 24–35. ISBN: 9781450318945.
- [248] Andre Mühlenbrock et al. "Fast and Robust Registration of Multiple Depth-Sensors and Virtual Worlds". In: *2021 International Conference on Cyberworlds (CW)*. 2021, pp. 41–48.
- [249] Élie Michel, Arnaud Emilien, and Marie-Paule Cani. "Generation of Folded Terrains from Simple Vector Maps". In: *Eurographics*. 2015.
- [250] Microsoft. *Azure Kinect DK depth camera documentation*. 2022. <https://docs.microsoft.com/en-us/azure/kinect-dk/depth-camera>. Last accessed 6. September 2022.
- [251] Dongbo Min, Jiangbo Lu, and Minh N. Do. "Depth Video Enhancement Based on Weighted Mode Filtering". In: *IEEE Transactions on Image Processing* 21.3 (2012), pp. 1176–1190.
- [252] Mark R. Mine. *Virtual Environment Interaction Techniques*. Tech. rep. 1995.
- [253] R. Mur-Artal and J. D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (Oct. 2017), pp. 1255–1262. ISSN: 1941-0468.
- [254] F Kenton Musgrave, Craig E Kolb, and Robert S Mace. "The synthesis and rendering of eroded fractal terrains". In: *ACM Siggraph Computer Graphics*. Vol. 23(2). ACM. 1989, pp. 41–50.
- [255] Aunoy K Mutasim, Anil Ufuk Batmaz, and Wolfgang Stuerzlinger. "Pinch, Click, or Dwell: Comparing Different Selection Techniques for Eye-Gaze-Based Pointing in Virtual Reality". In: *ACM Symposium on Eye Tracking Research and Applications*. ETRA '21 Short Papers. Virtual Event, Germany: Association for Computing Machinery, 2021.
- [256] R. A. Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. Oct. 2011, pp. 127–136.
- [257] Brian J. Nguyen et al. "Evaluation of Virtual Reality for Detection of Lung Nodules on Computed Tomography". In: *Tomography (Ann Arbor, Mich.)* 4.4 (Dec. 2018), pp. 204–208.

- [258] Wang Ning et al. "MUSICAL: Multi-Scale Image Contextual Attention Learning for Inpainting". In: Aug. 2019, pp. 3748–3754.
- [259] Timothy J Nokes-Malach, J Elizabeth Richey, and Soniya Gadgil. "When is it better to learn together? Insights from research on collaborative learning". In: *Educational Psychology Review* 27 (2015), pp. 645–656.
- [260] Kristine Nowak. "Defining and Differentiating Copresence, Social Presence and Presence as Transportation". In: (Nov. 2001).
- [261] Kristine Nowak and Frank Biocca. "The Effect of the Agency and Anthropomorphism on Users' Sense of Telepresence, Copresence, and Social Presence in Virtual Environments". In: *Presence Teleoperators and Virtual Environments* 12 (Oct. 2003), pp. 481–494.
- [262] Matthias Oberhauser et al. "What's Real About Virtual Reality Flight Simulation?: Comparing the Fidelity of a Virtual Reality With a Conventional Flight Simulation Environment". In: *Aviation Psychology and Applied Human Factors* 8 (Mar. 2018), pp. 22–34.
- [263] Catherine S. Oh, Jeremy N. Bailenson, and Gregory F. Welch. "A Systematic Review of Social Presence: Definition, Antecedents, and Implications". In: *Frontiers in Robotics and AI* 5 (2018).
- [264] Sergio Orts et al. "Holoportation: Virtual 3D Teleportation in Real-time". In: Oct. 2016.
- [265] Paulo Paiva et al. "SimCEC: A Collaborative VR-Based Simulator for Surgical Teamwork Education". In: *Computers in Entertainment* 16 (Apr. 2018), pp. 1–26.
- [266] Ye Pan and Anthony Steed. "The impact of self-avatars on trust and collaboration in shared virtual environments". In: *PLOS ONE* 12 (Dec. 2017), e0189078.
- [267] Emmanouil Panagiotou and Eleni Charou. "Procedural 3D terrain generation using Generative Adversarial Networks". In: *arXiv preprint arXiv:2010.06411* (2020).
- [268] Giuseppe Papari, Nasiru Idowu, and Trond Varslot. "Fast Bilateral Filtering for Denoising Large 3D Images". In: *IEEE Transactions on Image Processing* 26.1 (2017), pp. 251–261.
- [269] Mathias Parger et al. "Human Upper-Body Inverse Kinematics for Increased Embodiment in Consumer-Grade Virtual Reality". In: *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*. VRST '18. Tokyo, Japan: Association for Computing Machinery, 2018.
- [270] Jeeseung Park and Younggeun Kim. "Styleformer: Transformer based Generative Adversarial Networks with Style Vector". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8973–8982.
- [271] The Insight Partners. *Augmented reality (AR), virtual reality (VR), and mixed reality (MR) market size worldwide in 2021 and 2028 [Graph]*. In: Statista. 2022. URL: <https://www.statista.com/statistics/591181/global-augmented-virtual-reality-market-size/> (visited on 01/25/2023).
- [272] Harshada Patel, Michael Pettitt, and John R. Wilson. "Factors of collaborative working: A framework for a collaboration model". In: *Applied Ergonomics* 43.1 (2012), pp. 1–26.

- [273] Emilee Patrick et al. "Using a Large Projection Screen as an Alternative to Head-Mounted Displays for Virtual Environments". In: *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands, 2000, 478–485. ISBN: 1581132166.
- [274] Nicolas Pavie et al. "Procedural Texture Synthesis by Locally Controlled Spot Noise". In: May 2016.
- [275] Fabrizio Pece, Jan Kautz, and Tim Weyrich. "Adapting Standard Video Codecs for Depth Streaming". In: Jan. 2011, pp. 59–66.
- [276] Jordan Peck. *FastNoise SIMD*. Software library for noise. 2016. <https://github.com/Auburns/FastNoiseSIMD>. Last accessed 23 May 2019.
- [277] Fernando Pereira et al. "Point cloud coding: A privileged view driven by a classification taxonomy". In: *Signal Processing: Image Communication* 85 (Apr. 2020), p. 115862.
- [278] Ken Perlin. "An Image Synthesizer". In: *SIGGRAPH Computer Graphics* 19.3 (1985), pp. 287–296.
- [279] Ken Perlin. "Improving Noise". In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. ACM, 2002, pp. 681–682.
- [280] Michael Pfandler et al. "Virtual reality-based simulators for spine surgery: a systematic review". In: *The Spine Journal* 17.9 (2017), pp. 1352–1363. ISSN: 1529-9430.
- [281] Ken Pfeuffer et al. "Gaze + Pinch Interaction in Virtual Reality". In: *Proceedings of the 5th Symposium on Spatial User Interaction*. SUI '17. Association for Computing Machinery, 2017, 99–108.
- [282] Kuang Ping and Luo Dingli. "Conditional Convolutional Generative Adversarial Networks Based Interactive Procedural Game Map Generation". In: *Advances in Information and Communication*. Ed. by Kohei Arai, Supriya Kapoor, and Rahul Bhatia. Springer International Publishing, 2020, pp. 400–419.
- [283] Filipi Pires, Carlos Costa, and Paulo Dias. "On the Use of Virtual Reality for Medical Imaging Visualization". In: *Journal of Digital Imaging* 34 (2021), pp. 1034–1048.
- [284] Johanna Pirker. "The Potential of Virtual Reality for Aerospace Applications". In: *2022 IEEE Aerospace Conference (AERO)*. 2022, pp. 1–8.
- [285] Thammathip Piumsomboon et al. "Mini-Me: An Adaptive Avatar for Mixed Reality Remote Collaboration". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, 1–13.
- [286] Bernhard Preim, Patrick Saalfeld, and Christian Hansen. "Virtual and Augmented Reality for Educational Anatomy". In: *Digital Anatomy: Applications of Virtual, Mixed and Augmented Reality*. Springer, 2021, pp. 299–324.
- [287] Aniruddha Prithul, Isayas Adhanom, and eelke folmer. "Teleportation in Virtual Reality; A Mini-Review". In: *Frontiers in Virtual Reality* 2 (Oct. 2021).
- [288] E. Prodromou et al. "A Multi-User Virtual Reality Application For Visualization And Analysis In Medical Imaging". In: *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*. 2020, pp. 795–800.

- [289] Yeshwanth Pulijala et al. "Effectiveness of Immersive Virtual Reality in Surgical Training - A Randomized Control Trial". In: *Journal of Oral and Maxillofacial Surgery* 76 (Oct. 2017).
- [290] Christian Felix Purps, Simon Janzer, and Matthias Wölfel. "Reconstructing Facial Expressions of HMD Users for Avatars in VR". In: *ArtsIT, Interactivity and Game Creation*. Ed. by Matthias Wölfel, Johannes Bernhardt, and Sonja Thiel. Springer International Publishing, 2022, pp. 61–76.
- [291] Zhen Qin et al. "Image inpainting based on deep learning: A review". In: *Displays* 69 (2021), p. 102028. ISSN: 0141-9382.
- [292] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).
- [293] Irene Rae et al. "A Framework for Understanding and Designing Telepresence". In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '15. Vancouver, BC, Canada: Association for Computing Machinery, 2015, 1552–1566. ISBN: 9781450329224.
- [294] Eric Ragan et al. "Studying the Effects of Stereo, Head Tracking, and Field of Regard on a Small-Scale Spatial Judgment Task". In: *IEEE transactions on visualization and computer graphics* 19 (Aug. 2012).
- [295] Jussi Rantala et al. "Comparison of Controller-Based Locomotion Techniques for Visual Observation in Virtual Reality". In: *Multimodal Technologies and Interaction* 5.7 (2021), p. 31. ISSN: 2414-4088.
- [296] Anke V. Reinschluessel et al. "Virtual Reality for Surgical Planning – Evaluation Based on Two Liver Tumor Resections". In: *Frontiers in Surgery* 9 (2022).
- [297] Jorge Revelles et al. "An Efficient Parametric Algorithm for Octree Traversal". In: (May 2000).
- [298] César Iván Aguilar Reyes et al. "An Adaptive Virtual Reality-Based Training System for Pilots". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 66.1 (2022), pp. 1962–1966.
- [299] Jean Pierre Richa et al. "AdaSplats: Adaptive Splatting of Point Clouds for Accurate 3D Modeling and Real-Time High-Fidelity LiDAR Simulation". In: *Remote Sensing* 14.24 (2022). ISSN: 2072-4292.
- [300] Simon Riches et al. "Factors Affecting Sense of Presence in a Virtual Reality Social Environment: A Qualitative Study". In: *Cyberpsychology, behavior and social networking* 22 4 (2019), pp. 288–292.
- [301] Riccardo Rigon et al. "Optimal Channel Networks - a Framework for the Study of River Basin Morphology". In: *Water Resources Research* 29 (June 1993), pp. 1635–1646.
- [302] Ruben Rodriguez Torrado et al. "Bootstrapping Conditional GANs for Video Game Level Generation". In: *IEEE Conference on Games (CoG)*. 2020, pp. 41–48.
- [303] Greg Roelofs and Richard Koman. *PNG: The Definitive Guide*. O'Reilly & Associates, Inc., 1999. ISBN: 1565925424.
- [304] Tae Hoon Roh et al. "Virtual dissection of the real brain: integration of photographic 3D models into virtual reality and its effect on neurosurgical resident education". In: *Neurosurgical Focus* 51.2 (2021), E16.

- [305] Edgar Rojas et al. "Telementoring in Leg Fasciotomies via Mixed-Reality: Clinical Evaluation of the STAR Platform". In: *Military Medicine* 185 (Jan. 2020), pp. 513–520.
- [306] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.
- [307] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Springer International Publishing, 2015, pp. 234–241.
- [308] David L. Rosgen. "A classification of natural rivers". In: *Catena* 22 (1994), pp. 169–199.
- [309] Juergen Rossmann and Michael Schluse. "Virtual Robotic Testbeds: A Foundation for e-Robotics in Space, in Industry - And in the Woods". In: *2011 Developments in E-systems Engineering*. 2011, pp. 496–501.
- [310] Daniel Roth and Marc Erich Latoschik. "Construction of the Virtual Embodiment Questionnaire (VEQ)". In: *IEEE Transactions on Visualization and Computer Graphics* 26.12 (2020), pp. 3546–3556.
- [311] Daniel Roth et al. "Real-time Mixed Reality Teleconsultation for Intensive Care Units in Pandemic Situations". In: *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. Mar. 2021, pp. 693–694.
- [312] Elby Roy, Mahmoud M. Bakr, and Roy George. "The need for virtual reality simulators in dental education: A review". In: *The Saudi Dental Journal* 29.2 (2017), pp. 41–47. ISSN: 1013-9052.
- [313] Daniel Ruijters and Anna Vilanova. "Optimizing GPU volume rendering". In: *Journal of WSCG* 14.1-3 (2006), pp. 9–+.
- [314] Tijana Ruzic and Aleksandra Pizurica. "Context-Aware Patch-Based Image Inpainting Using Markov Random Field Modeling". In: *IEEE Transactions on Image Processing* 24.1 (2015), pp. 444–456.
- [315] Rafael Sacks, Amotz Perlman, and Ronen Barak. "Construction safety training using immersive virtual reality". In: *Construction Management and Economics* 31 (Sept. 2013), pp. 1005–1017.
- [316] Timothy Sanders and Paul Cairns. "Time Perception, Immersion and Music in Videogames". In: *Proceedings of the 24th BCS Interaction Specialist Group Conference*. 2010, 160–167.
- [317] Shyam Prathish Sargunam and Eric D. Ragan. "Evaluating Joystick Control for View Rotation in Virtual Reality with Continuous Turning, Discrete Turning, and Field-of-View Reduction". In: *Proceedings of the 3rd International Workshop on Interactive and Spatial Computing*. IWISC '18. Richardson, Texas: Association for Computing Machinery, 2018, 74–79.
- [318] Sukla Satapathy and Rajiv Ranjan Sahay. "Robust depth map inpainting using superpixels and non-local Gauss-Markov random field prior". In: *Signal Processing: Image Communication* 98 (2021), p. 116378.
- [319] Peter Scarfe and Andrew Glennerster. "The Science Behind Virtual Reality Displays". In: *Annual Review of Vision Science* 5 (Sept. 2019).

- [320] D. Scharstein and C. Pal. "Learning Conditional Random Fields for Stereo". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. June 2007, pp. 1–8.
- [321] Ingrid Scholl et al. "MedicVR". In: *Bildverarbeitung für die Medizin 2019*. Springer Fachmedien Wiesbaden, 2019, pp. 152–157.
- [322] Danny Schott et al. "A VR/AR Environment for Multi-User Liver Anatomy Education". In: *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 2021, pp. 296–305.
- [323] Danny Schott et al. "A VR/AR Environment for Multi-User Liver Anatomy Education". In: *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 2021, pp. 296–305.
- [324] Thomas Schöps et al. "Large-scale outdoor 3D reconstruction on a mobile device". In: *Computer Vision and Image Understanding 157 (2017)*. Large-Scale 3D Modeling of Urban Indoor or Outdoor Scenes from Images and Range Scans, pp. 151–166. ISSN: 1077-3142.
- [325] Christoph Schröder et al. "DynCam: A Reactive Multithreaded Pipeline Library for 3D Telepresence in VR". In: *Proc. of the 20th ACM Virtual Reality International Conference (VRIC 2018)*. ACM. 2018. ISBN: 978-1-4503-5381-6.
- [326] Christoph Schröder et al. "DynCam: A Reactive Multithreaded Pipeline Library for 3D Telepresence in VR". In: *Proceedings of the Virtual Reality International Conference - Laval Virtual*. VRIC '18. Laval, France: Association for Computing Machinery, 2018. ISBN: 9781450353816.
- [327] Thomas Schubert, Frank Friedmann, and Holger Regenbrecht. "The Experience of Presence: Factor Analytic Insights". In: *Presence: Teleoperators and Virtual Environments 10.3 (June 2001)*, pp. 266–281.
- [328] Thomas W. Schubert. "A New Conception of Spatial Presence: Once Again, with Feeling". In: *Communication Theory 19.2 (Apr. 2009)*, pp. 161–187. ISSN: 1050-3293.
- [329] S. Schwarz et al. "Emerging MPEG Standards for Point Cloud Compression". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9.1 (Mar. 2019)*, pp. 133–148. ISSN: 2156-3365.
- [330] Valentin Schwind et al. "Using Presence Questionnaires in Virtual Reality". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, 1–12. ISBN: 9781450359702.
- [331] Hanna Söderholm et al. "The potential impact of 3D telepresence technology on task performance in emergency trauma care". In: Nov. 2007, pp. 79–88.
- [332] Yusuf Sermet and Ibrahim Demir. "GeospatialVR: A web-based virtual reality framework for collaborative environmental simulations". In: *Computers & Geosciences 159 (2022)*, p. 105010. ISSN: 0098-3004.
- [333] S. Shahriyar et al. "Lossless depth map coding using binary tree based decomposition and context-based arithmetic coding". In: *2016 IEEE International Conference on Multimedia and Expo (ICME)*. July 2016, pp. 1–6.
- [334] Mingwen Shao et al. "Multi-scale generative adversarial inpainting network based on cross-layer attention transfer mechanism". In: *Knowledge-Based Systems 196 (2020)*, p. 105778.



- [335] Ling Shen et al. "Single-Shot Semantic Image Inpainting with Densely Connected Generative Networks". In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. 2019, pp. 1861–1869.
- [336] William Sherman and Alan Craig. *Understanding Virtual Reality—Interface, Application, and Design*. Jan. 2002. ISBN: 1558603530.
- [337] William R Sherman and Alan B Craig. *Understanding virtual reality: Interface, application, and design*. Morgan Kaufmann, 2018.
- [338] Dev Yashpal Sheth et al. "Unsupervised Deep Video Denoising". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 1759–1768.
- [339] John Short, Ederyn Williams, and Bruce Arthur Christie. "The social psychology of telecommunications". In: 1976.
- [340] Anjali Singal, Agam Bansal, and Priti Chaudhary. *Cadaverless anatomy: Darkness in the times of pandemic Covid-19*. 2020.
- [341] Mel Slater. "A Note on Presence Terminology". In: *Presence Connect* 3 (Jan. 2003).
- [342] Mel Slater. "Immersion and the illusion of presence in virtual reality". In: *British Journal of Psychology* 109 (Mar. 2018).
- [343] Mel Slater. "Immersion and the illusion of presence in virtual reality". In: *British journal of psychology* 109.3 (2018), pp. 431–433.
- [344] Mel Slater. "Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 364 (2009), pp. 3549–3557.
- [345] Mel Slater, Bernhard Spanlang, and David Corominas. "Simulating Virtual Environments within Virtual Environments as the Basis for a Psychophysics of Presence". In: *ACM Trans. Graph.* 29.4 (July 2010).
- [346] Mel Slater et al. "A Separate Reality: An Update on Place Illusion and Plausibility in Virtual Reality". In: *Frontiers in Virtual Reality* 3 (June 2022), p. 914392.
- [347] Ruben M Smelik et al. "A survey on procedural modelling for virtual worlds". In: *Computer Graphics Forum*. Vol. 33(6). Wiley Online Library. 2014, pp. 31–50.
- [348] Sun Wha Song et al. "Clinical utility of three-dimensional facial computed tomography in the treatment of nasal bone fractures: a new modality involving an air-bone view with a volume rendering technique". In: *Indian journal of otolaryngology and head and neck surgery : official publication of the Association of Otolaryngologists of India* 65.Suppl 2 (Aug. 2013), pp. 210–215.
- [349] Yanan Song, Weiming Shen, and Kunkun Peng. "A novel partial point cloud registration method based on graph attention network". In: *The Visual Computer* 39 (Feb. 2022).
- [350] Ryan J. Spick, Peter Cowling, and James Alfred Walker. "Procedural Generation using Spatial GANs for Region-Specific Learning of Elevation Data". In: *2019 IEEE Conference on Games (CoG)*. 2019, pp. 1–8.
- [351] ryan rs spick and james walker. "Realistic and Textured Terrain Generation Using GANs". In: *Proceedings of the 16th ACM SIGGRAPH European Conference on Visual Media Production*. CVMP '19. London, United Kingdom: Association for Computing Machinery, 2019. ISBN: 9781450370035.

- [352] Misha Sra and Chris Schmandt. "MetaSpace: Full-body Tracking for Immersive Multiperson Virtual Reality". In: Nov. 2015, pp. 47–48.
- [353] Jos Stam. "Real-time fluid dynamics for games". In: *Proceedings of the Game Developer Conference*. 2003.
- [354] J.-L. Starck, M. Elad, and D.L. Donoho. "Image decomposition via the combination of sparse representations and a variational approach". In: *IEEE Transactions on Image Processing* 14.10 (2005), pp. 1570–1582.
- [355] Ondrej Stava et al. "Interactive Terrain Modeling Using Hydraulic Erosion." In: July 2008, pp. 201–210.
- [356] Anthony Steed et al. "An 'In the Wild' Experiment on Presence and Embodiment using Consumer Virtual Reality Equipment". In: *IEEE Transactions on Visualization and Computer Graphics* 22.4 (2016), pp. 1406–1414.
- [357] Tomasz F. Stepinski and Chaitanya Bagaria. "Segmentation-Based Unsupervised Terrain Classification for Generation of Physiographic Maps". In: *IEEE Geoscience and Remote Sensing Letters* 6.4 (2009), pp. 733–737.
- [358] Vladimiro Sterzentsenko et al. "Self-Supervised Deep Depth Denoising". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.
- [359] Robin Strak et al. "Comparison Between Video-Mediated and Asymmetric 3D Teleconsultation During a Preclinical Scenario". In: *Proceedings of Mensch Und Computer 2021. MuC '21*. Ingolstadt, Germany: Association for Computing Machinery, 2021, 227–235.
- [360] J. Sturm et al. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012.
- [361] Po-Chang Su, Ju Shen, and Muhammad Usman Rafique. "RGB-D Camera Network Calibration and Streaming for 3D Telepresence in Large Environment". In: *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*. 2017, pp. 362–369.
- [362] Zhuo Su et al. "RobustFusion: Robust Volumetric Performance Reconstruction Under Human-Object Interactions From Monocular RGBD Stream". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 45 (2021), pp. 6196–6213.
- [363] G. J. Sullivan et al. "Overview of the High Efficiency Video Coding (HEVC) Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pp. 1649–1668. ISSN: 1558-2205.
- [364] Xuebin Sun et al. "A Novel Point Cloud Compression Algorithm Based on Clustering". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2132–2139.
- [365] Zhonghua Sun. "Multislice CT angiography in abdominal aortic aneurysm treated with endovascular stent grafts: Evaluation of 2D and 3D visualisations". In: *Biomedical Imaging and Intervention Journal* 3 (Oct. 2007).
- [366] Ivan E Sutherland et al. "The ultimate display". In: *Proceedings of the IFIP Congress*. Vol. 2. 506-508. New York. 1965, pp. 506–508.
- [367] Roman Suvorov et al. "Resolution-Robust Large Mask Inpainting With Fourier Convolutions". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2022, pp. 2149–2159.

- [368] Susumu Tachi. "From 3D to VR and further to telexistence". In: *2013 23rd International Conference on Artificial Reality and Telexistence (ICAT)*. 2013, pp. 1–10.
- [369] François-Xavier Talgorn and Farès Belhadj. "Real-Time Sketch-Based Terrain Generation". In: June 2018, pp. 13–18.
- [370] Yifu Tao et al. "3D Lidar Reconstruction with Probabilistic Depth Completion for Robotic Navigation". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2022, pp. 5339–5346.
- [371] Matias Tassano, Julie Delon, and Thomas Veit. "FastDVDnet: Towards Real-Time Deep Video Denoising Without Flow Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [372] Chia-Chi Teng et al. "Interactive Augmented Live Virtual Reality Streaming: A Health Care Application". In: June 2018, pp. 143–147. ISBN: 978-1-4503-6389-1.
- [373] Soon Teoh. "RiverLand: An Efficient Procedural Modeling System for Creating Realistic-Looking Terrains". In: Nov. 2009, pp. 468–479. ISBN: 978-3-642-10330-8.
- [374] Joern Teuber et al. "VaMEx-VTB – A Modular Virtual Testbed for Multimodal Autonomous Planetary Missions". In: *Proceeding of the 70th International Astronautical Congress. International Astronautical Congress (IAC-2019), October 21-25, Washington DC, DC, United States*. 2019.
- [375] Nadia Thalmann and Daniel Thalmann. "Virtual humans: Thirty years of research, what next?" In: *The Visual Computer* 21 (Dec. 2005), pp. 997–1015.
- [376] D. Thanou, P. A. Chou, and P. Frossard. "Graph-Based Compression of Dynamic 3D Point Cloud Sequences". In: *IEEE Transactions on Image Processing* 25.4 (Apr. 2016), pp. 1765–1778. ISSN: 1941-0042.
- [377] Santawat Thanyadit et al. "Substituting Teleportation Visualization for Collaborative Virtual Environments". In: *Symposium on Spatial User Interaction. SUI '20. Virtual Event, Canada: Association for Computing Machinery*, 2020. ISBN: 9781450379434.
- [378] Jörn Thieling, Manuel Mathar, and Jürgen Roßmann. "Automated generation of virtual road scenarios for efficient tests of driver assistance systems". In: *2017 IEEE AUTOTESTCON*. 2017, pp. 1–9.
- [379] Jan-Noël Thon. "Immersion revisited: on the value of a contested concept". In: *Extending Experiences. Structure, Analysis and Design of Computer Game Player Experience*. Lapland University Press, 2008, pp. 29–43.
- [380] Balasaravanan Thoravi Kumaravel et al. "Loki: Facilitating Remote Instruction of Physical Tasks Using Bi-Directional Mixed-Reality Telepresence". In: Oct. 2019, pp. 161–174. ISBN: 978-1-4503-6816-2.
- [381] Michal Tölgyessy et al. "Evaluation of the Azure Kinect and Its Comparison to Kinect V1 and Kinect V2". In: *Sensors* 21 (Jan. 2021), p. 413.
- [382] Theodoros Toghias et al. "Virtual reality environment for industrial robot control and path design". In: *Procedia CIRP* 100 (2021). 31st CIRP Design Conference 2021 (CIRP Design 2021), pp. 133–138. ISSN: 2212-8271.

- [383] Nikolaos Tsamitros et al. "Virtual Reality-Based Treatment Approaches in the Field of Substance Use Disorders". In: *Current Addiction Reports* 8 (Sept. 2021), pp. 1–9.
- [384] D. Tschumperle and R. Deriche. "Vector-valued image regularization with PDEs: a common framework for different applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.4 (2005), pp. 506–517.
- [385] Chenxi Tu et al. "Compressing continuous point cloud data using image compression methods". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 1712–1719.
- [386] Huawei Tu et al. "Crossing-Based Selection with Virtual Reality Head-Mounted Displays". In: *CHI '19*. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, 1–14.
- [387] Dimitrios Ververidis, Spiros Nikolopoulos, and Ioannis Kompatsiaris. "A Review of Collaborative Virtual Reality Systems for the Architecture, Engineering, and Construction Industry". In: *Architecture* 2.3 (2022), pp. 476–496.
- [388] Ryan J. Vitacion and Li Liu. "Procedural Generation of 3D Planetary-Scale Terrains". In: *2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*. 2019, pp. 70–77.
- [389] Vanessa Volz et al. "Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '18. Kyoto, Japan: Association for Computing Machinery, 2018, 221–228.
- [390] Georgios Voulgaris, Ioannis Mademlis, and Ioannis Pitas. "Procedural Terrain Generation Using Generative Adversarial Networks". In: *2021 29th European Signal Processing Conference (EUSIPCO)*. 2021, pp. 686–690.
- [391] Gregory K Wallace. "The JPEG still picture compression standard". In: *IEEE transactions on consumer electronics* 38.1 (1992), pp. xviii–xxxiv.
- [392] Thomas Waltemate et al. "The Impact of Avatar Personalization and Immersion on Virtual Body Ownership, Presence, and Emotional Response". In: *IEEE Transactions on Visualization and Computer Graphics* 24.4 (2018), pp. 1643–1652.
- [393] Thomas Waltemate et al. "The Impact of Latency on Perceptual Judgments and Motor Performance in Closed-Loop Interaction in Virtual Reality". In: *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*. VRST '16. Munich, Germany: Association for Computing Machinery, 2016, 27–35.
- [394] Cheng Yao Wang et al. "Again, Together: Socially Reliving Virtual Reality Experiences When Separated". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, 1–12. ISBN: 9781450367080.
- [395] Huixiang Wang et al. "Application of an innovative computerized virtual planning system in acetabular fracture surgery: A feasibility study". In: *Injury* 47.8 (2016), pp. 1698–1701.
- [396] Ruoqing Wang, Sufei Li, and Ercan E. Kuruoglu. "A Novel Algorithm for Image Denoising Based on Unscented Kalman Filtering". In: *Int. J. Inf. Commun. Technol.* 5.3/4 (June 2013), 343–353. ISSN: 1466-6642.

- [397] Sinong Wang et al. "Linformer: Self-Attention with Linear Complexity". In: *ArXiv abs/2006.04768* (2020).
- [398] Tzu-Yang Wang et al. "Effect of Body Representation Level of an Avatar on Quality of AR-Based Remote Instruction". In: *Multimodal Technologies and Interaction* 4.1 (2020).
- [399] Z. Wang, E.P. Simoncelli, and A.C. Bovik. "Multiscale structural similarity for image quality assessment". In: *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. 2003, 1398–1402 Vol.2.
- [400] O. Wasenmüller, M. Meyer, and D. Stricker. "Augmented Reality 3D Discrepancy Check in Industrial Applications". In: *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Sept. 2016, pp. 125–134.
- [401] M. J. Weinberger, G. Seroussi, and G. Sapiro. "LOCO-I: a low complexity, context-based, lossless image compression algorithm". In: *Proceedings of Data Compression Conference - DCC '96*. Mar. 1996, pp. 140–149.
- [402] M. J. Weinberger, G. Seroussi, and G. Sapiro. "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS". In: *IEEE Transactions on Image Processing* 9.8 (Aug. 2000), pp. 1309–1324. ISSN: 1941-0042.
- [403] Tim Weissker and Bernd Froehlich. "Group Navigation for Guided Tours in Distributed Virtual Environments". In: *IEEE Transactions on Visualization and Computer Graphics* 27.5 (2021), pp. 2524–2534.
- [404] René Weller et al. "Effects of immersion and navigation agency in virtual environments on emotions and behavioral intentions". In: *Frontiers in Virtual Reality* 3 (2022). ISSN: 2673-4192.
- [405] René Weller et al. "LenSelect: Object Selection in Virtual Environments by Dynamic Object Scaling". In: *Frontiers in Virtual Reality* 2 (2021), p. 70.
- [406] Rene Weller, Benjamin Brennecke, and Gabriel Zachmann. "Redirected walking in virtual reality with auditory step feedback". In: *The Visual Computer* 38 (July 2022), pp. 1–12.
- [407] Rene Weller et al. "VR-Interactions for Planning Planetary Swarm Exploration Missions in VaMEx-Vtb". In: *2021 IEEE Aerospace Conference (50100)*. 2021, pp. 1–11.
- [408] Stephan Wenninger et al. "Realistic Virtual Humans from Smartphone Videos". In: *Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology*. VRST '20. Association for Computing Machinery, 2020.
- [409] Thomas Whelan et al. "Real-time large-scale dense RGB-D SLAM with volumetric fusion". In: *The International Journal of Robotics Research* 34.4-5 (2015), pp. 598–626.
- [410] Robert H Whittaker. *Communities and Ecosystems*. Macmillan Company, 1975.
- [411] T. Wiegand et al. "Overview of the H.264/AVC video coding standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (July 2003), pp. 560–576. ISSN: 1558-2205.
- [412] Andrew D. Wilson. "Fast Lossless Depth Image Compression". In: *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. ISS '17. Brighton, United Kingdom: Association for Computing Machinery, 2017, 100–105. ISBN: 9781450346917.

- [413] Preston Tunnell Wilson et al. "VR Locomotion: Walking > Walking in Place > Arm Swinging". In: *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*. VRCAI '16. Zhuhai, China: Association for Computing Machinery, 2016, 243–249. ISBN: 9781450346924.
- [414] Isabell Wohlgenannt, Alexander Simons, and Stefan Stieglitz. "Virtual reality". In: *Business & Information Systems Engineering* 62.5 (2020), pp. 455–461.
- [415] Ryan Wongsa. *PyTorch Implementation of the paper: Image Inpainting for Irregular Holes Using Partial Convolutions*. 2020. <https://github.com/ryanwongsa/Image-Inpainting>.
- [416] Yuanjie Wu et al. "Using a Fully Expressive Avatar to Collaborate in Virtual Reality: Evaluation of Task Performance, Presence, and Attraction". In: *Frontiers in Virtual Reality* 2 (Apr. 2021).
- [417] Andreas Wulff-Jensen et al. "Deep Convolutional Generative Adversarial Network for Procedural 3D Landscape Generation Based on DEM". In: *6th EAI International Conference on Arts and Technology, Interactivity & Game Creation*. Jan. 2018, pp. 85–94. ISBN: 978-3-319-76907-3.
- [418] Nannan Xi and Juho Hamari. "Shopping in virtual reality: A literature review and future agenda". In: *Journal of Business Research* 134 (2021), pp. 37–58.
- [419] Jinsheng Xiao et al. "Blind video denoising via texture-aware noise estimation". In: *Computer Vision and Image Understanding* 169 (2018), pp. 1–13. ISSN: 1077-3142.
- [420] Chaohao Xie et al. "Image Inpainting With Learnable Bidirectional Attention Maps". In: Oct. 2019, pp. 8857–8866.
- [421] Xu Xie et al. "VRGym: A Virtual Testbed for Physical and Interactive AI". In: *Proceedings of the ACM Turing Celebration Conference - China*. ACM TURC '19. Association for Computing Machinery, 2019. ISBN: 9781450371582.
- [422] Wenge Xu et al. "Evaluation of Text Selection Techniques in Virtual Reality Head-Mounted Displays". In: *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2022, pp. 131–140.
- [423] Zhaoyi Yan et al. "Shift-Net: Image Inpainting via Deep Feature Rearrangement". In: *European Conference on Computer Vision*. 2018.
- [424] Eun Kyoung Yang, Jee Hyun Lee, and C. Hun Lee. "Virtual reality environment-based collaborative exploration of fashion design". In: *CoDesign* 0.0 (2023), pp. 1–19.
- [425] Su Yang, Shishuo Xu, and Wei Huang. "3D Point Cloud for Cultural Heritage: A Scientometric Survey". In: *Remote Sensing* 14.21 (2022). ISSN: 2072-4292.
- [426] Shouwen Yao. "Autonomous-driving vehicle test technology based on virtual reality". In: *The Journal of Engineering* 2018 (Aug. 2018).
- [427] Raymond Yeh et al. "Semantic Image Inpainting with Deep Generative Models". In: July 2017, pp. 6882–6890.
- [428] Mary K. Young, John J. Rieser, and Bobby Bodenheimer. "Dyadic Interactions with Avatars in Immersive Virtual Environments: High Fiving". In: *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception*. SAP '15. Tübingen, Germany: Association for Computing Machinery, 2015, 119–126. ISBN: 9781450338127.

- [429] Difeng Yu et al. "Target Selection in Head-Mounted Display Virtual Reality Environments". In: *J. Univers. Comput. Sci.* 24 (2018), pp. 1217–1243.
- [430] Jiahui Yu et al. "Free-Form Image Inpainting With Gated Convolution". In: Oct. 2019, pp. 4470–4479.
- [431] Kevin Yu et al. "Avatars for Teleconsultation: Effects of Avatar Embodiment Techniques on User Perception in 3D Asymmetric Telepresence". In: *IEEE Transactions on Visualization and Computer Graphics* PP (Aug. 2021), pp. 1–1.
- [432] Kevin Yu et al. "Duplicated Reality for Co-located Augmented Reality Collaboration". In: *IEEE Transactions on Visualization and Computer Graphics* 28.5 (2022), pp. 2190–2200.
- [433] Yingchen Yu et al. "Diverse Image Inpainting with Bidirectional and Autoregressive Transformers". In: *Proc. of the 29th ACM Int. Conf. on Multimedia* (2021).
- [434] R.C. Zeleznik et al. "Pop through button devices for VE navigation and interaction". In: *Proceedings IEEE Virtual Reality 2002*. 2002, pp. 127–134.
- [435] Chenyang Zhang, Teng Huang, and Qiang Zhao. "A New Model of RGB-D Camera Calibration Based On 3D Control Field". In: *Sensors* 19 (Nov. 2019), p. 5082.
- [436] Huijie Zhang et al. "Synthetic Modeling Method for Large Scale Terrain Based on Hydrology". In: *IEEE Access* 4 (2016), pp. 6238–6249.
- [437] Jian Zhang et al. "Authoring Multi-style Terrain with Global-to-Local Control". In: *Graphical Models* 119 (Nov. 2022).
- [438] Qiuwen Zhang et al. "Low-complexity depth map compression in HEVC-based 3D video coding". In: *EURASIP Journal on Image and Video Processing* 2015.1 (2015), p. 2.
- [439] Xiaobo Zhang et al. "Image inpainting based on deep learning: A review". In: *Information Fusion* 90 (Sept. 2022).
- [440] Yinda Zhang and Thomas Funkhouser. "Deep Depth Completion of a Single RGB-D Image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.
- [441] Hanqing Zhao et al. "Extraction of Terraces on the Loess Plateau from High-Resolution DEMs and Imagery Utilizing Object-Based Image Analysis". In: *ISPRS International Journal of Geo-Information* 6.6 (2017).
- [442] Howard Zhou et al. "Terrain Synthesis from Digital Elevation Models". In: *IEEE transactions on visualization and computer graphics* 13 (Aug. 2007), pp. 834–48.