FACULTY OF DIGITAL ENGINEERING
Computer Graphics Systems Group

Real-Time Rendering Techniques for Massive 3D Point Clouds

Echtzeit-Rendering-Techniken für massive 3D-Punktwolken

Dissertation
in partial fulfillment for the academic degree
"doctor rerum naturalium"
(Dr. rer. nat.)
in Computer Science

Hasso Plattner Institute // Faculty of Digital Engineering // University of Potsdam

submitted by
Sören Discher

Supervision:
Prof. Dr. Jürgen Döllner
Chair: Computer Graphics Systems

Potsdam,
December 6, 2022

Supervisor and 1st reviewer:
Prof. Dr. Jürgen Döllner
Hasso-Plattner-Institut/Universität Potsdam

2nd reviewer:
Prof. Dr. Raffaele de Amicis
Oregon State University

3rd reviewer:
Prof. Dr. Hartmut Asche
Hasso-Plattner-Institut/Universität Potsdam

Date of final exam: 13.07.2023

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-60164
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-601641

Contents

Abstract

Today, point clouds are among the most important categories of spatial data, as they constitute digital 3D models of the as-is reality that can be created at unprecedented speed and precision. However, their unique properties, i.e., lack of structure, order, or connectivity information, necessitate specialized data structures and algorithms to leverage their full precision. In particular, this holds true for the interactive visualization of point clouds, which requires to balance hardware limitations regarding GPU memory and bandwidth against a naturally high susceptibility to visual artifacts.

This thesis focuses on concepts, techniques, and implementations of robust, scalable, and portable 3D visualization systems for massive point clouds. To that end, a number of rendering, visualization, and interaction techniques are introduced, that extend several basic strategies to decouple rendering efforts and data management: First, a novel visualization technique that facilitates context-aware filtering, highlighting, and interaction within point cloud depictions. Second, hardware-specific optimization techniques that improve rendering performance and image quality in an increasingly diversified hardware landscape. Third, natural and artificial locomotion techniques for nausea-free exploration in the context of state-of-the-art virtual reality devices. Fourth, a framework for web-based rendering that enables collaborative exploration of point clouds across device ecosystems and facilitates the integration into established workflows and software systems.

In cooperation with partners from industry and academia, the practicability and robustness of the presented techniques are showcased via several case studies using representative application scenarios and point cloud data sets. In summary, the work shows that the interactive visualization of point clouds can be implemented by a multi-tier software architecture with a number of domain-independent, generic system components that rely on optimization strategies specific to large point clouds. It demonstrates the feasibility of interactive, scalable point cloud visualization as a key component for distributed IT solutions that operate with spatial digital twins, providing arguments in favor of using point clouds as a universal type of spatial base data usable directly for visualization purposes.

Zusammenfassung

Punktwolken gehören heute zu den wichtigsten Kategorien räumlicher Daten, da sie digitale 3D-Modelle der Ist-Realität darstellen, die mit beispielloser Geschwindigkeit und Präzision erstellt werden können. Ihre einzigartigen Eigenschaften, d.h. das Fehlen von Struktur-, Ordnungs- oder Konnektivitätsinformationen, erfordern jedoch spezielle Datenstrukturen und Algorithmen, um ihre volle Präzision zu nutzen. Insbesondere gilt dies für die interaktive Visualisierung von Punktwolken, die es erfordert, Hardwarebeschränkungen in Bezug auf GPU-Speicher und -Bandbreite mit einer naturgemäß hohen Anfälligkeit für visuelle Artefakte in Einklang zu bringen.

Diese Arbeit konzentriert sich auf Konzepte, Techniken und Implementierungen von robusten, skalierbaren und portablen 3D-Visualisierungssystemen für massive Punktwolken. Zu diesem Zweck wird eine Reihe von Rendering-, Visualisierungs- und Interaktionstechniken vorgestellt, die mehrere grundlegende Strategien zur Entkopplung von Rendering-Aufwand und Datenmanagement erweitern: Erstens eine neuartige Visualisierungstechnik, die kontextabhängiges Filtern, Hervorheben und Interaktion innerhalb von Punktwolkendarstellungen erleichtert. Zweitens hardwarespezifische Optimierungstechniken, welche die Rendering-Leistung und die Bildqualität in einer immer vielfältigeren Hardware-Landschaft verbessern. Drittens natürliche und künstliche Fortbewegungstechniken für eine übelkeitsfreie Erkundung im Kontext moderner Virtual-Reality-Geräte. Viertens ein Framework für webbasiertes Rendering, das die kollaborative Erkundung von Punktwolken über Geräteökosysteme hinweg ermöglicht und die Integration in etablierte Workflows und Softwaresysteme erleichtert.

In Zusammenarbeit mit Partnern aus Industrie und Wissenschaft wird die Praxistauglichkeit und Robustheit der vorgestellten Techniken anhand mehrerer Fallstudien aufgezeigt, die repräsentative Anwendungsszenarien und Punktwolkendatensätze verwenden. Zusammenfassend zeigt die Arbeit, dass die interaktive Visualisierung von Punktwolken durch eine mehrstufige Softwarearchitektur mit einer Reihe von domänenunabhängigen, generischen Systemkomponenten realisiert werden kann, die auf Optimierungsstrategien beruhen, die speziell für große Punktwolken geeignet sind. Sie demonstriert die Machbarkeit einer interaktiven, skalierbaren Punktwolkenvisualisierung als Schlüsselkomponente für verteilte IT-Lösungen, die mit räumlichen digitalen Zwillingen arbeiten, und liefert Argumente für die Verwendung von Punktwolken als universelle Art von räumlichen Basisdaten, die direkt für Visualisierungszwecke verwendet werden können.

Chapter 1

Introduction

This chapter gives an overview of this thesis' motivation and objective goals, the essence of its technical contribution as well as the broader impacts the presented research is expected to have –and already had– on the application domain.

1.1 Motivation and Goals

Technologies for capturing, processing, and visualizing spatial data are key for digitization in a growing number of application areas as they provide means to quickly and accurately create, modify, explore, and visualize digital 3D models of spatial objects, environments, or phenomena. In particular, point clouds are among the most important categories of spatial data, as they enable us, for example, to captured snapshots of the as-is reality by a generic representation that consists of a discrete set of three-dimensional points. Point clouds do not have structure, order, or connectivity information, so specialized data structures and algorithms are necessary to use point clouds in applications, tools and systems. For example, interactive visualization of point clouds must take into account that graphics processors provide limited memory and bandwidth. A commonly applied and proven strategy to nonetheless render massive data sets with billions of points is the combined use of *level-of-detail* (LoD) representations and out-of-core rendering algorithms. While rendering techniques implementing that strategy constitute a well-covered research area [69, 141, 186, 66], considerably less research has been aimed at the refinement of these basic rendering techniques into professionally usable 3D visualization systems. In particular, there exists a gap in knowledge regarding three areas: First, assisting users in conducting use case specific visualization and exploration tasks, e.g., by means of visual filtering and highlighting. Second, portability of interactive point cloud visualizations across an ever-more diversified hardware landscape, e.g., by means of hardware-specific render optimization strategies, interaction, and locomotion techniques. Third, integration of point cloud visualizations into established workflows and software systems, e.g., by enabling collaborative data access and manipulation via standardized interfaces. While considerable advances have been made in these areas with respect to mesh-based 3D models [58, 155, 177, 126], it remains largely unknown to what extent those contributions may also be effectively applied to point clouds. As a remedy, 3D meshes are still widely considered to be the more easily handleable, more flexible and, thus, preferable representation of spatial data for visualization purposes. For example,

even current iterations of popular *geographic information systems* (GIS) often default to using point clouds as input to derive generalized 3D meshes, rather than visualizing them directly – especially for massive data sets containing billions of points. Thus, the full potential and density of point clouds is rarely leveraged by established applications, tools and systems in today's market.

This thesis contributes towards the long-term goal of establishing point clouds as a universal type of spatial base data usable directly for analysis and visualization purposes. Concentrating on concepts, techniques and implementations for real-time rendering of point clouds, it investigates how a number of strategies to control rendering efforts and to optimize data management can be applied to achieve robust, scalable, and portable 3D visualization systems for massive point clouds. In particular, this thesis addresses the aforementioned specific gaps in knowledge:

- It presents a new visualization technique for context-aware filtering, highlighting, and interaction that incorporates thematic data layers such as temporal information or surface categories derived via deep learning; this facilitates and speeds up a number of common visualization and exploration tasks for point clouds.
- It introduces several optimization techniques improving rendering performance and image quality; this is key to implement natural and artificial locomotion techniques as part of immersive, nausea-free exploration using state-of-the-art *Virtual Reality* (VR) devices.
- It presents a framework for web-based rendering that enables interactive visualization of massive point clouds even on low-end mobile devices; this is key for IT solutions that build on service-oriented computing and that allow various stakeholders to access and use applications based on point cloud data.

All contributions in this thesis have been evaluated against representative application scenarios and point cloud data sets, showcasing their adaptability to changing requirements and how they scale with respect to the point cloud size. The results clearly indicate the practicability and robustness of the presented techniques and provide arguments in favor of establishing point clouds as a universal type of spatial base data usable directly for visualization purposes. This is further underlined by several case studies conducted together with partners from industry and academia that apply these techniques to build specialized tools and applications addressing specific real-world use cases (see Chapter 6).

1.2 Intellectual Merit and Problem Statement

Precise and detailed *digital 3D models* of real-world objects, environments, and phenomena have become highly relevant for a large variety of geospatial [53] and non-geospatial application domains [189]. As an example, *"3D digitization"* [120] is commonly applied to document and preserve cultural and natural heritage of any size, ranging from individual artifacts [193] over sprawling building complexes [80] to entire landscapes [51].

Forming *"a permanent record for current and future conservation, research and educational purposes"* [151], digital 3D models in this context allow users to explore and inspect artifacts and sites they would not be able to access in person, e.g., due to ongoing conservatory work or for political, legal, and safety reasons, while ensuring that *"the information of the shape and appearance of an object is not lost in case of damage"* [68]. In various industry sectors, such as the manufacturing, automotive industry or architecture, digital 3D models of real-world buildings and infrastructures play a crucial role, serving as base data to build *digital twins* [62]: Constituting *"an integrated multiphysics, multiscale, probabilistic simulation of a complex product, which functions to mirror the life of its corresponding twin"* [171], such digital twins are deemed *"a key enabler for the digital transformation"* [87]. As an example, they may *"support operators' understanding and decision-making"* [122], enable *"predictive maintenance for critical [automotive] subsystems"* [97], or *"assist researchers do experiments and testing of medicines in virtual patients to reduce risks and costs"* [92]. In the *architecture, engineering, and construction* (AEC) industry, digital 3D models and digital twins alike can facilitate the design, construction, operation, and maintenance of large-scale building and infrastructure projects, for which typically a multitude of stakeholders needs to be coordinated [20]: In a process commonly referred to as *building information modeling* (BIM), digital 3D models are generated for all assets acquired over the lifetime of a project, including buildings and built infrastructure (e.g., pipelines, railways) [33] as well as movable parts such as vehicles, building interiors, or outdoor furniture [60]. Generated models are synchronized with a database serving as a central repository, allowing stakeholders to easily browse, locate, and monitor a project's inventory, thus *"engendering collaboration"* [100] and potentially *"improving design, construction and maintenance practices"* [65]. Recent years have seen this concept being adapted for other domains, such as facility management [169] or cultural and natural heritage preservation [95]. On a larger scale, it can also be applied to large-scale infrastructure networks, cities and metropolitan areas, or even countries, rather than individual sites [192, 27]. Here, the primary focus is typically on inventorying ground heights, buildings, vegetation, or city furniture (e.g., street signs, lamp posts), possibly automated via AI technology as a means *"to transcend explicit geospatial modelling"* [54]. In turn, the corresponding digital 3D models allow us to facilitate and automatize applications in diverse areas such as land surveying and landscape architecture [180], urban planning and development [26], environmental monitoring [147], and disaster management [61].

Technological advances in in-situ and remote sensing technology have opened up efficient ways to digitize real-world or built objects and sites by generating point clouds that can be either used as is or further converted to 3D meshes [25] (Figure 1.1). Two fundamental approaches are used to obtain point clouds: *Light Detection and Ranging* (LiDAR)[1] [56] and a combination of high resolution imagery and photogrammetric analyses [15]. Unlike approaches based on mesh-based or primitive-based 3D geometric modeling, point cloud acquisition involves almost no manual postprocessing, that is,

[1]Sometimes also referred to as *Laser Detection and Ranging (LADAR)*.

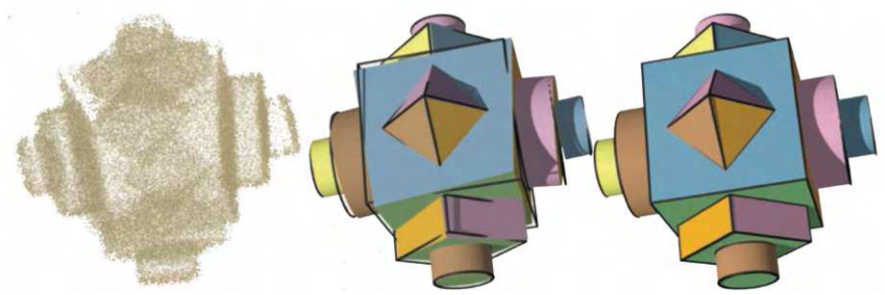


Figure 1.1: *Surface reconstruction algorithms allow to derive 3D meshes even from noisy point clouds. Figure has been taken from [25].*

the raw data results from an automated process. As an example, state-of-the-art laser scanners can capture several thousands of points per second [144], generating millimeter-precise point clouds of individual sites, such as the "Teatro Olimpico" in Vicenza, Italy with an approximated cubic volume of 7700m³ [184], acquired within a few working days [72]. Larger areas are covered by attaching the scanning device to a moving vehicle, such as cars or *unmanned aircraft systems (UAS)*, resulting in point clouds that contain several billion points and terabytes of raw data [90, 98]. Due to its increased affordability and effectiveness, communities worldwide have started to intensify the use of in-situ and remote sensing technology by conducting scans more regularly (e.g., once a year) and by combining different capturing methods (e.g., aerial laser scanning [106] and mobile mapping [128]), thus, establishing point clouds "as the national core data for geo-information" [176]. Similarly, 3D scanning technology is being integrated into an ever-increasing number of consumer electronics such as video game consoles [74] or mobile devices [45], which "lowers the entry barrier for both private users and industrial users to digitize objects" [179].

Existing tools and applications are often limited in their ability to handle point clouds, that - unlike 3D meshes - both inherently have some representational blur and lack order or connectivity. Traditional analysis algorithms for geodata often struggle with these properties as they commonly rely on explicitly defined connectivity information. Existing rendering techniques for point clouds are often quite straightforward, for example, applying a uniform pixel size and render style to each point and, therefore, are prone to visual artifacts such as holes or visual clutter, which severely limits perception of structures, interaction, and navigation [162] (Figure 1.2). As a remedy, existing applications and systems (e.g., GIS) tend to utilize point clouds primarily as input data to derive more easily handleable 3D meshes (e.g., 3D city models, 3D terrain models) [25], rather than to operate on the point data directly. Depending on the use case, this may require a time-consuming and only semi-automatic process that does not scale for massive data sets, especially if precision, density, and data quality of the derived 3D meshes need to be maximized. That issue is only aggravated by ever-improving scanning hardware and novel, cheaper, and easier-to-use carrier systems, which will result in more massive point clouds in the future. Thus, there is a strong demand to efficiently store, manage

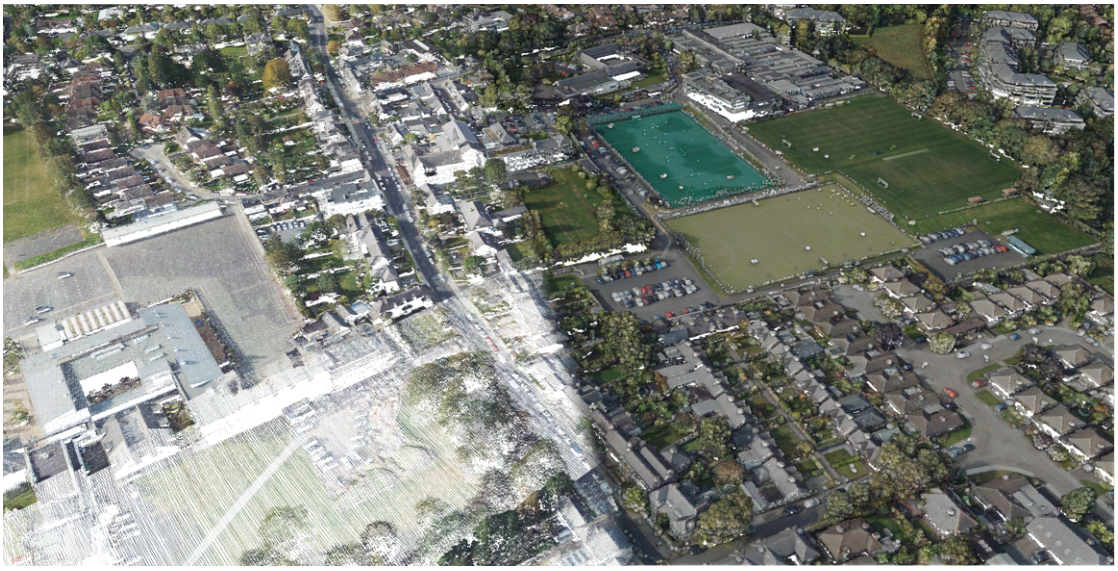


Figure 1.2: *Effect of different rendering techniques on the perception of structures within point clouds depictions: Uniform pixel size and rendering style per point (left). Adaptive pixel size per point and highlighted depth differences (right). Data courtesy of Ordnance Survey Ireland.*

and process as well as interactively explore point clouds to take advantage of their full potential and to provide an unfiltered, detailed representation of captured sites.

This thesis concentrates on rendering and interaction techniques for massively sized point clouds that may contain an arbitrary number of additional data layers, e.g., per-point color values or thematic information (see Section 2.1). The constraints and conditions for those techniques include:

- Massive raw data needs to be processed (e.g., several terabytes), that typically surpasses available main memory and GPU memory capacities by a wide margin.
- A number of additional data layers computed by pre-processing steps and analyses needs to be considered by real-time rendering; for example, visual filters and highlighting techniques are required for that reason.
- A broad range of hardware platforms with varying processing and graphics capabilities need to be supported (e.g., low-end mobile devices, high-end desktop computers and emerging VR devices), necessitating the need for different data management approaches, rendering strategies, and interaction techniques.

Interactively visualizing massive data sets with billions of points requires to decouple rendering efforts and data management. This can be achieved by subdividing point clouds into small, representative subsets that are suitable for real-time rendering and that can be selected dynamically, e.g., based on the current view and available memory [71, 156]. Traditionally, such out-of-core algorithms are designed for a specific use case and a

comparatively small group of users: GIS-like applications running on high-end desktop computers with the complete data being directly accessible via local storage. Even in current iterations of popular GIS (e.g., ESRI ArcGIS[2] or QGIS[3]) the underlying processing paradigms are still largely built around this use case — not least to ensure compatibility to older software versions. However, recent technological trends and breakthroughs have established new ways to present and interact with point clouds, rendering traditional desktop applications more and more outdated:

- Following the trend of device ecosystems, mobile devices, tablets, or web-based applications are replacing the traditional high-end workstation. Their lack of local storage necessitates web-based rendering techniques that enable streaming of the required data from a central server. Combining out-of-core and web-based rendering techniques for point clouds raises additional challenges (e.g., regarding network traffic, data security, or load balancing) and has therefore become an active field of research [98, 48] that is further discussed in Chapter 5.

- Recent years have seen the emergence of sophisticated VR devices (e.g., HTC Vive, Oculus Rift), granting users the perception of being physically present within a digital scene [133, 113, 19]. However, such immersive visualizations not only rely on significantly higher frame rates than traditional non-immersive visualizations to avoid feelings of motion sickness, they are also highly sensitive to any kind of visual artifacts (e.g., stitching) that may break the immersion. Hence, additional rendering techniques have to be designed, implemented, and evaluated that optimize existing non-immersive rendering systems for massive point clouds with respect to two conflicting goals: An improved visual quality of render artifacts and an increased rendering performance. Likewise, specialized locomotion techniques need to be provided and evaluated since navigating 3D virtual environments can easily result in orientation loss or motion sickness. Details are discussed in Chapter 4.

Rendering performance and visual quality are just two aspects that need to be considered when designing systems for the interactive exploration of point clouds: Users of such systems are usually motivated by specific visualization or exploration tasks, e.g., identifying all newly constructed buildings within an area. If used to their full potential, additional data layers (e.g., temporal information, thematic data) can notably speed up such tasks. However, current rendering systems for point clouds rarely utilize such data layers for more than straightforward gradients or classification-based color schemes [159, 140, 190]. Therefore, context-aware filtering, highlighting, and interaction techniques that incorporate additional data layers to facilitate the recognition and inspection of objects, semantics, and temporal changes within point cloud depictions are discussed in Chapter 3.

[2] https://www.esri.com/en-us/arcgis/about-arcgis/overview
[3] https://www.qgis.org/en/site/

Chapter 2

Foundations and Background

This chapter provides a detailed overview of the research field, introducing and discussing terminology and related work relevant to the subsequent chapters. The chapter is based in parts on the author's scientific publications in [2], [1], [3], [5], and [7].

The following sections are structured as follows: Relevant terminology for the remainder of this work is defined in Section 2.1, followed by a short characterization of different acquisition techniques in Section 2.2 In Section 2.3 the concept of a multi-tier software architecture enabling the efficient management, processing, and visualization of point clouds is introduced. This concept is followed up by Section 2.4, discussing the advantages and disadvantages of common spatial data structures, and Section 2.5, introducing a modular service-oriented processing pipeline based on out-of-core and GPU-based processing to enable a seamless integration into existing workflows and systems. Section 2.6 concludes with a discussion of the state-of-the-art regarding visualization and interaction techniques for point clouds, in particular regarding immersive and web-based visualizations.

2.1 Terminology

In the context of this thesis, the terms **point cloud**, **enriched point cloud**, and **4D point cloud** will be defined as detailed below.

2.1.1 Point Clouds

With the term **point cloud**, we refer in the following to an unstructured collection of discrete points in a three-dimensional Euclidean Space. It is denoted by $\mathbf{P} = p_0, p_1, ..., p_N$ with points $p \in \mathbf{R}^3$. The metaphor *cloud* draws the analogy between the unorganized water particles in a cloud and the 3D points in a point cloud.

Point clouds, partially due to the acquisition technologies, are characterized by the following properties:

- **Simplicity**: Point clouds exclusively rely on point sampling. They do not ensure a specific density or a regular point distribution within the cloud. They do not model nor reflect topological surface properties, which must be reconstructed if needed.
- **Genericity**: Point clouds can represent three-dimensional objects in a generic way as they approximate the surface or boundaries by point samples.

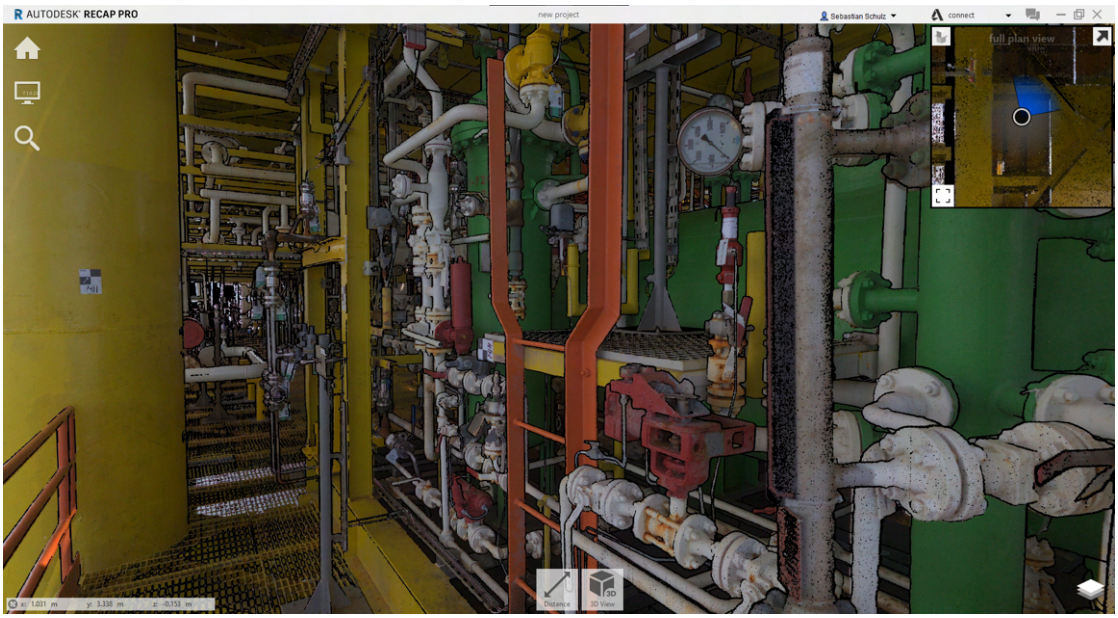


Figure 2.1: *Point cloud of an engine room, representing millimeter-scale surface details.*

Figure 2.2: *Fusion of several point clouds representing different sites and objects within the inner city of Landshut, Germany, at varying scales. Data was captured and provided by Elektra Solar and Illustrated Architecture.*

- **Density**: The average distance between points, i.e., the density of a point cloud can vary drastically, ranging from micrometers to several meters.
- **Scalability**: Point clouds apply to object scales, ranging from micrometer-scale surface details [21] to plate-tectonic surface displacements [108, 119] (Figure 2.1, Figure 2.2).
- **Massiveness**: Due to the scale in terms of number of points that typical acquisition processes produce, point clouds can be considered as "big data", e.g., having millions or billions of points in a single scan.
- **Invariance**: Point clouds are invariant regarding permutations. As they do not define any order or structure among the points, the points are strictly unorganized.

Point clouds are increasingly becoming a universal, cross-application category of spatial data and therefore form basic data for a growing number of geospatial [54] and non-geospatial applications, systems, and services [183]. In the context of this thesis, point clouds may be of arbitrary origin (e.g., acquired via mobile mapping, terrestrial, or aerial acquisition campaigns), density, scale (e.g., representing individual objects up to entire cities) and massiveness, representing a snapshot of the captured surface at a specific point in time.

2.1.2 Enriched Point Clouds

With the term **enriched point cloud** we refer in the following to a point cloud that has been enhanced with additional data layers, i.e., per-point attributes. Typical examples of such data layers can be categorized as follows:

- **Geometric Information**: Data layers that characterize the relationship between neighboring points. Common examples are scalar values describing the height of a point in relation to points within its proximity as well as per-point normals approximating the surface of the local point proximity [103].
- **Graphical Information**: Data layers that can be directly applied to change a point's graphical representation and stylization. In particular, this refers to color or color-infrared values that have been extracted from images captured alongside the point cloud.
- **Surface Material Properties**: Data layers that provide information about the material properties of the surface represented by a point, e.g., the reflectivity of a surface.
- **Surface Category and Object Information**: Data layers that categorize points with respect to the type of surface represented by them. This may range from a coarse categorization of points into building, ground, and vegetation [139] to highly specific categories [125], e.g., different building parts or archetypes of vegetation.

In some cases, points may even be assigned to individual objects that are part of a surface category, e.g., individual trees [109].

- **Thematic Information**: Data layers that provide georeferenced thematic information about a captured site (e.g., cadastre data, statistical information) or objects within (e.g., sensor data) that has been projected on the point cloud.

Compared to point clouds without additional data layers, enriched point clouds facilitate the design and implementation of task and domain-specific tools and applications [124, 115]. In the context of this thesis, the handling and visualization of enriched point clouds will be the main focus.

2.1.3 4D Point Clouds

With the term **4D point cloud** we refer in the following to data sets that combine multiple point clouds of the same site or object, taken at different points in time. The individual point clouds are typically enriched with additional data layers. In particular, they contain *time stamps*, describing the date of data collection, as well as *change information*, i.e., scalar values describing on a per-point basis the degree of change related to a given reference point cloud.

2.2 Data Acquisition

Approaches for acquiring point clouds are manifold, whereby in general they allow us to capture real-world objects. In particular, the *surfaces* of such objects are captured at all scales, ranging from tiny surface details (e.g., varnish control) over small objects (e.g., sorting fruits) to complete buildings, cities or even countries (e.g., environmental monitoring). In literature, two different kinds of sensors are typically distinguished:

- *Active* sensors (e.g., LiDAR, radar, stripe projection systems), which emit electromagnetic radiation to directly measure the distance between surface and sensor, generating so-called range data [57].

- *Passive* sensors (e.g., aerial cameras, digital cameras, hyperspectral radiometers), which rely on natural radiation, most notably sunlight, to generate series of images [137] that may be used as input for dense image matching algorithms to derive 3D information [148].

Both, active and passive sensors, can capture individual objects with point densities of up to a few micrometers [136, 23]. The resolution, in general, depends on the distance between sensor and captured surface. On a larger scale, point clouds of rooms, buildings, or facilities can be generated by capturing scenes at key positions within the site in question [135]. By attaching the sensors to moving vehicles such as cars, trains, *Unmanned Aircraft Systems* (UAS), planes, helicopters, or satellites, data for large-scale areas such

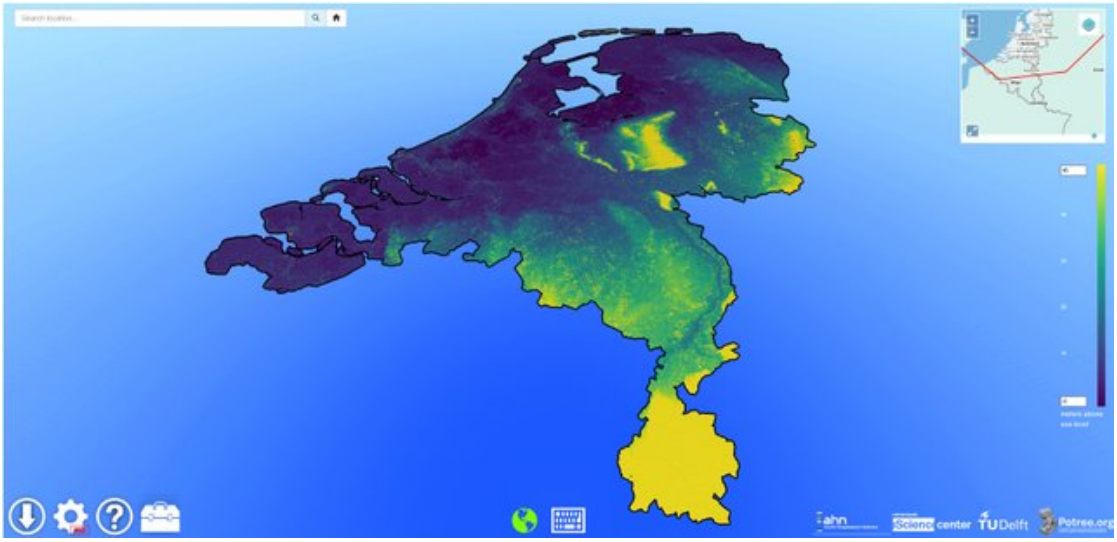


Figure 2.3: *Nation-wide point cloud of the Netherlands. Figure has been taken from [98].*

as infrastructure networks, cities, or countries can be efficiently collected (Figure 2.3), albeit at reduced point densities [114].

Each acquisition approach comes with specific advantages and disadvantages, affecting its suitability for different use cases (Table 2.1). In general, passive sensors tend to be more affordable, portable, and easier to use than their active counterparts, as evidenced by their frequent integration into state-of-the-art consumer electronics [84] and UAS. Furthermore, image-based methods allow for potentially higher point densities, e.g., up to 400 points/m^2 for aerial photographs as in contrast to typically 1-25 points/m^2 for aerial laser scans [137]. However, the quality of the resulting point clouds is significantly influenced by surface materials (e.g., shininess, texturedness) and image properties (e.g., shadows, color variety, depth of field). This can be especially noticeable when capturing glassy surfaces, where range-based approaches tend to generate much cleaner point clouds [137]. With respect to performance, passive sensors collect data faster; however, a computation-intense post-processing of the generated images is required to compute 3D points, whereas active sensors provide those directly [135]. Table 2.2 provides an overview of several common acquisition systems and their specific characteristics regarding typical point density and acquisition costs.

In practice, different acquisition systems are frequently used in parallel: Advanced driver assistance systems, for example, combine several active and passive sensors to observe a car's immediate surroundings [89]. Similarly, digitization projects for cultural heritage frequently involve stationary acquisition systems for close quarter sections as well as mobile ones for open areas [63, 185]. The resulting data sets may cover large areas while still preserving tiny details, amounting to hundreds of billions of points and terabytes of raw data.

Table 2.1: *Typical scale of commonly used acquisition systems for point clouds.*

Acquisition System	Typical Scale
Airborne Laserscanning	Infrastructure networks urban + rural areas
Aerial Photography	Infrastructure networks, urban + rural areas
Mobile Mapping (rails, roads)	Infrastructure networks, urban areas
UAS	Buildings, facilities, infrastructure networks
Static Terrestrial Laserscanning	Indoor scenes, buildings, facilities
Smartphone cameras / DSLRs	Individual objects, indoor scenes, buildings
Depth Cameras	Individual objects, indoor scenes
Stripe-projection Systems	Individual objects, indoor scenes

Table 2.2: *Typical density and costs of commonly used acquisition systems for point clouds.*

Acquisition System	Typical Density (pts/m^2)	Costs
Airborne Laserscanning	1 - 25	very high
Aerial Photography	25 - 400	high
Mobile Mapping (rails, roads)	200 - 1,400	medium
UAS	500 - 6,000	medium
Static Terrestrial Laserscanning	4,000 - 20,000	medium
Smartphone cameras / DSLRs	4,000 - 40,000	low
Depth Cameras	4,000 - 20,000	low
Stripe-projection Systems	100,000 - 400,000	low

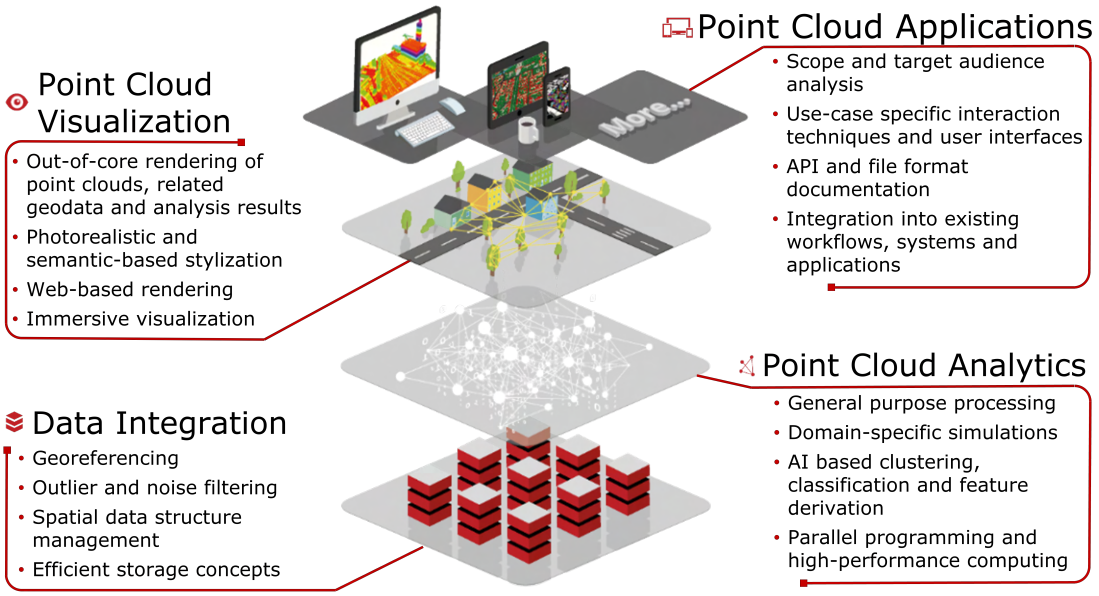


Figure 2.4: *Concept of a multi-tier software architecture providing functionality for data management, analytics, visualization, and end-user applications in the context of enriched point clouds.*

2.3 Software Architectures for Enriched Point Clouds

As outlined in Chapter 1, the improved affordability and robustness of state-of-the-art in-situ and remote sensing technology have led to point clouds becoming an increasingly relevant source of spatial information in corresponding application domains. However, the use of point clouds as a central data source poses a challenge for any IT system due to their inherent complexity and size. Therefore, from a software engineering perspective, multi-tier software architectures are required that should decouple the components for data management and storage, processing and analysis, visualization and exploration, and end-user applications. (Figure 2.4):

- **Data Integration Layer:** Includes components for the integration and quality control of point clouds from *heterogeneous* data sources (e.g., active or passive sensors, mobile or stationary acquisition systems) into a *homogeneous* spatial data model featuring efficient data storage concepts. This encompasses (1) georeferencing acquired data, i.e., associating each point with a concrete spatial location via well-defined geodetic reference systems, (2) filtering noise and outliers as well as (3) creating and updating spatial data structures that ensure an efficient access to arbitrary data subsets during subsequent operations.

- **Point Cloud Analytics Layer:** Includes components for general-purpose processing tasks (e.g., calculating per-point surface normals, mapping external color

information onto a point cloud), domain-specific simulations as well as clustering, classification, and feature derivation, frequently based on *Artificial Intelligence* (AI) concepts such as *Machine Learning* (ML) or *Deep Learning* (DL) [54]. As a prerequisite, such components may require additional data layers (i.e., geometric or thematic information), which can be either calculated on a per-point basis or queried from dedicated web services that are accessed via standardized protocols such as *Web Processing Services* (WPS) [105] or *Web Map Services* (WMS) [88]. The size of point clouds requires implementations of high-performance computing and parallel processing strategies, e.g., using spatial data structures provided by the *Data Integration* layer.

- **Point Cloud Visualization Layer:** Includes components for interactively visualizing point clouds, related geodata (e.g., digital terrain models of the area in question), and analysis data; this layer should operate on various platforms, ranging from low-end mobile devices over high end desktop computers to emerging VR systems. Each platform differs with respect to available memory capacities and graphics capabilities as well as requirements regarding visual quality and rendering speed, necessitating the use of vastly different rendering strategies. Furthermore, to facilitate the visual inspection of a point cloud‘s many data layers, semantics-driven rendering and stylization techniques must be provided that highlight different aspects of a data set.

- **Point Cloud Application Layer:** Includes components that combine functionality provided by the underlying layers into end-user applications. Scope and target audience of these applications may be vastly different, ranging from, e.g., extension modules for traditional desktop-based GIS aimed at professionals [31, 47], over web platforms facilitating the upload, storage and collaborative exploration of point clouds [123, 124], to VR-based presentations of historical sites meant to educate non-experts [116, 42]. Thus, the design and implementation of suitable interaction techniques and user interfaces that are customized for the corresponding use cases is a crucial aspect. Moreover, to facilitate their integration into existing workflows and systems, well-documented and openly accessible interfaces and file formats constitute a prerequisite for components of this layer.

The contributions provided by this thesis are related to the point cloud visualization layer and the point cloud application layer. Efficient spatial data structures and processing concepts, constitute the basis on which techniques of these layers operate and are therefore briefly discussed in the following sections.

2.4 Spatial Data Structures

For any non-trivial processing, analysis, or visualization technique, it is necessary that subsets of a point cloud can be spatially queried in a random and efficient manner. The data structures necessary for this require pre-processing of the point cloud, but in

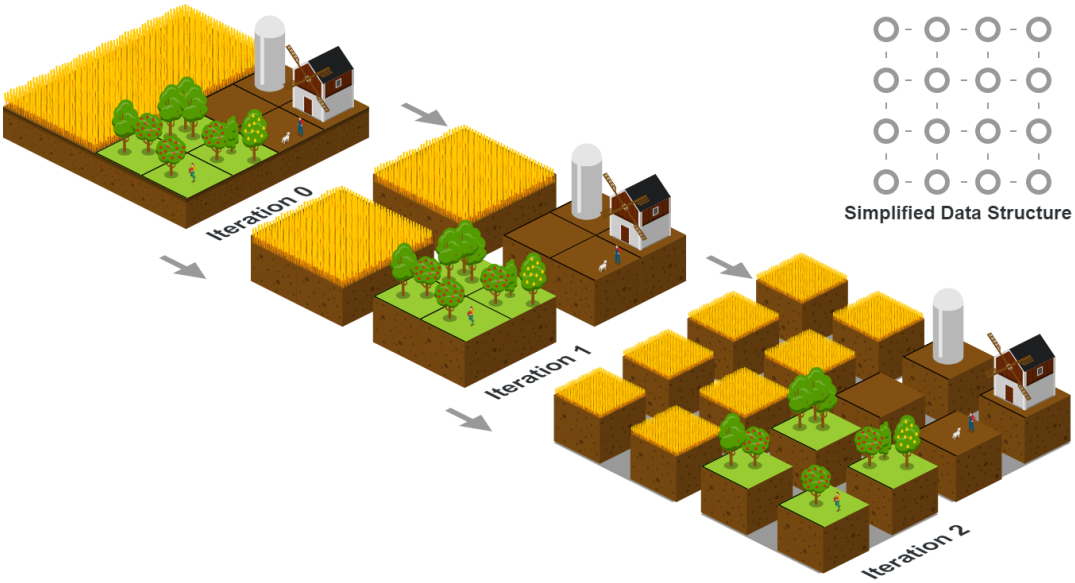


Figure 2.5: *Uniform grids split a given space into a fixed number of equally sized cells (or voxels), without applying any LoD concepts. While they are easy to manage, they are inefficient when the data density is non-uniformly distributed.*

particular the construction of LoD representations to spatially subdivide and structure the a priori unstructured points of the point cloud into smaller, more manageable subsets. Of the many options, the following spatial data structures are most commonly used in the context of point clouds:

2.4.1 Uniform Grids

A *uniform grid* is the simplest form of a spatial data structure [153, 13] as it simply divides a given space into a fixed number of equally size grid cells (Figure 2.5). Individual *cells* —or *voxels* in the case of a three-dimensional grid— are accessed via indices; LoD concepts are not applied. Due to their simplicity, uniform grids can be expanded and regressed very efficiently. However, varying point densities are not taken into account, making this spatial data structure inefficient for non-uniformly distributed data, since either a comparatively large number of cells would remain empty and essentially unused or some cells would hoard an impractical massive amount of points.

2.4.2 Quadtrees

A *quadtree* is defined as a tree structure that recursively subdivides a *two-dimensional area* into equally sized *quadrants* until a maximum number of points is reached for each leaf [154]. Its overall efficiency depends on the number of traversal operations needed to locate and extract a point or group of neighboring points, which in turn is primarily influenced by its balancedness. Thus, the tree depth should be as equal as possible across

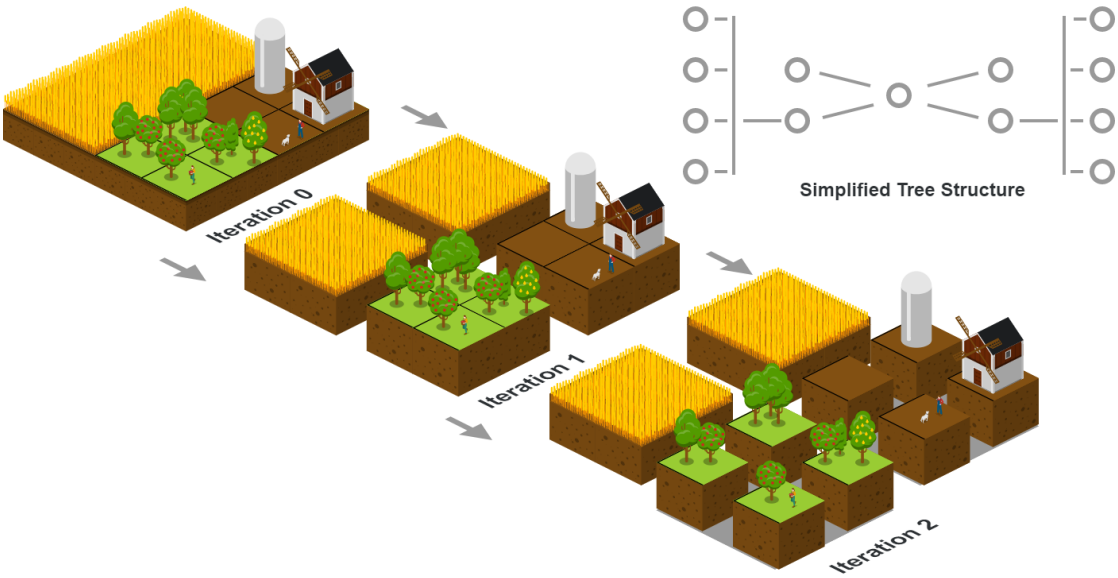


Figure 2.6: *Quadtrees recursively subdivide a two-dimensional area into a number of equally sized quadrants until the amount of details stored in each leaf reaches a predefined maximum. They are most efficient if the data is primarily distributed non-uniformly along two coordinate axes.*

all areas covered by the quadtree. As a remedy, quadtrees tend to be suited best in scenarios where the data is primarily distributed horizontally, which is typically true for aerial acquisitions (Figure 2.6). Even in scenarios with a less ideal point distribution however, they still tend to outperform uniform grids in terms of look-up speed and memory consumption as long as the point distribution is "somewhat" non-uniform. On the other hand, spatial expansion and regression of a quadtree is drastically more complex compared to uniform grids as such operations necessitate restructuring the whole tree structure.

2.4.3 Octrees

In contrast to a quadtree, an *octree* [102] constitutes a recursive subdivision of a *three-dimensional* space into equally sized *octants* (Figure 2.7). Again, the recursion stops once a minimal number of points for each leaf of the tree structure is reached. If the data is non-uniformly distributed both horizontally and vertically —as it is often the case for terrestrial or mobile mapping acquisitions— octrees are typically more balanced and thus more efficient to traverse than quadtrees. Otherwise however, many leaves will be sparsely populated or even completely empty, resulting in an unnecessarily high memory consumption. Finally, adding and removing points to and from an octree comes with caveats similar to those known from quadtrees: Such changes can be conducted with minimal effort unless the overall spatial bounds have to be expanded or regressed, which would require to restructure the tree.

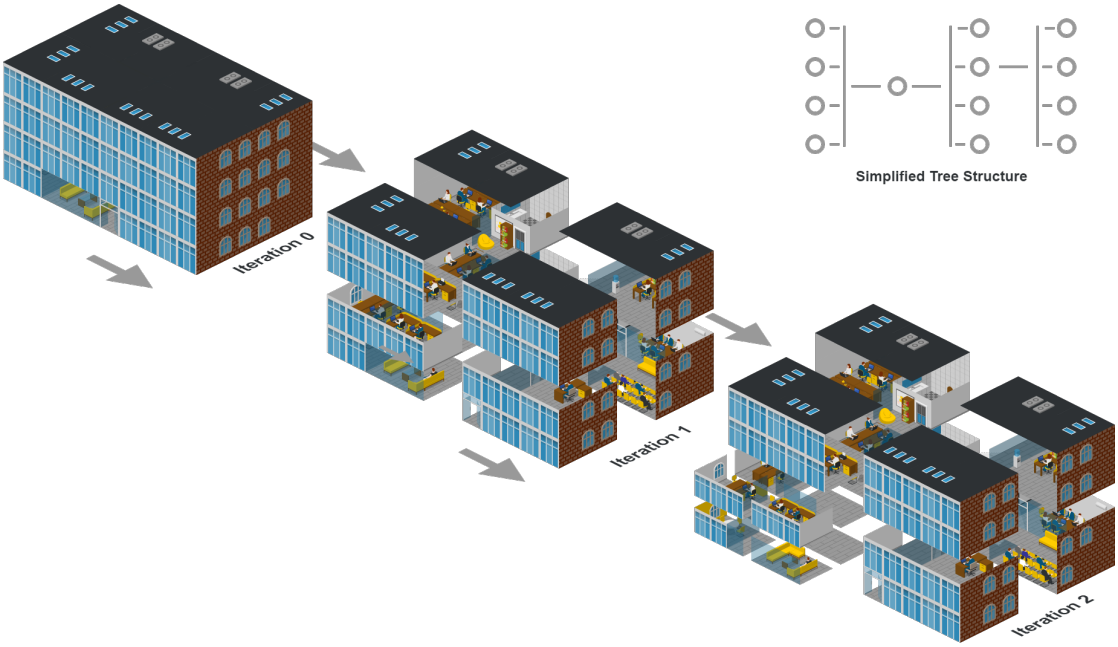


Figure 2.7: *Octrees recursively subdivide a three-dimensional space into a number of equally sized octants until the amount of details stored in each leaf reaches a predefined maximum. They are most efficient if the data is distributed non-uniformly along all three coordinate axes.*

2.4.4 Kd-Trees

As a generalization of quadtrees and octrees, *kd-trees* organize points in a k-dimensional space, allowing to create perfectly balanced tree structures independently of the data's spatial distribution [22]. Unlike quadtrees or octrees, kd-trees are *binary*, i.e., each inner node is split into two rather than four and eight subsets, respectively. Splitting planes applied during each iteration are not fixed but can instead be positioned freely alongside the respective coordinate axis (Figure 2.8). While this minimizes tree traversal times during data look-up, it also results in a significantly more complex construction and update process compared to data structures based on fixed splitting planes. Thus, the practical use of kd-trees is typically limited to static data sets that are not expected to be updated frequently.

2.4.5 Multi-Layered Data Structures

None of the described spatial data structures should be seen as one fits all solutions. On the contrary, many use cases stand to benefit from combining different approaches. As an example, large parts of a city may be scanned at lengthy intervals, whereas certain hotspots (e.g., active construction sites) need to be re-scanned and updated frequently. Here, maintaining separate spatial data structures for each of those sites that are combined into an overlaying uniform grid or a quadtree allows to integrate locally constrained updates without intense reordering of the entire spatial data structure

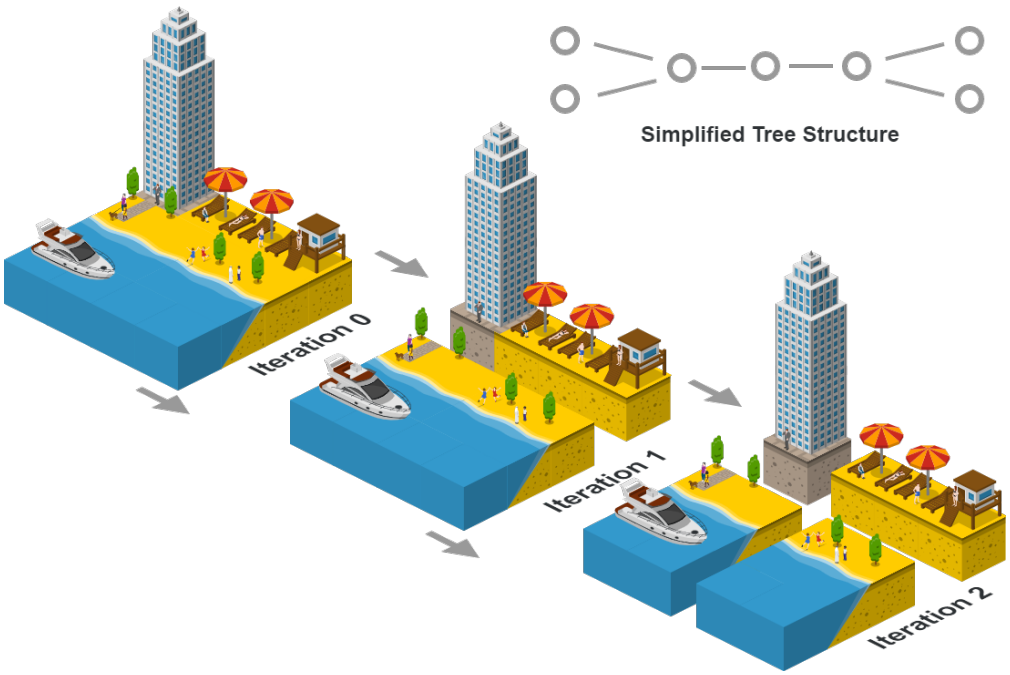


Figure 2.8: *Kd-Trees recursively subdivide a k-dimensional space via flexible splitting planes, that can be freely positioned alongside the coordinate axes. As a remedy, the resulting binary trees are perfectly balanced.*

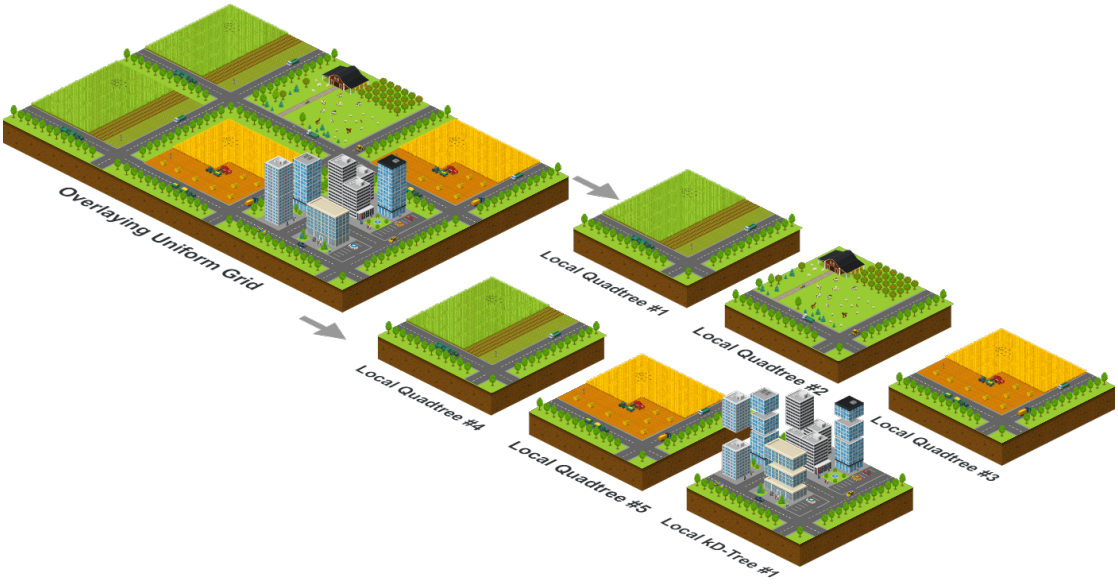


Figure 2.9: *In practice, many data sets benefit from so-called multi-layered approaches. Here, several spatial data structures are combined that each organize different types of sites (e.g., quadtrees for plain, rural areas and kd-trees for more convoluted, urban areas.*

(Figure 2.9). Furthermore, multi-layered data structures may be used to organize points based on multiple data layers. This will be further demonstrated in Chapter 3 by splitting semantically enriched point clouds based on each point's surface category (e.g., building, vegetation); for each surface category a separate kd-tree is maintained.

2.5 Point Cloud Analytics Concepts

Data and use case specific operations and analyses on point clouds typically combine several atomic processing tasks (e.g., determining a point's closest neighbor or aggregating attribute values within a point's proximity) that can be executed independently on a small area around each point [40, 30, 139, 142]. Therefore, the processing performance can be significantly increased by applying parallel computing concepts, either based on a CPU or a GPU. Furthermore, different processing tasks can be efficiently chained together by interleaving them: Instead of executing each task one at a time for the complete data set, processed subsets are immediately subjected to subsequent tasks. Since point clouds commonly exceed available capacities of main or GPU memory, these parallel computing concepts need to combined with out-of-core approaches that subdivide the overall data set into sufficiently small subsets.

A practical example that combines parallel computing and out-of-core concepts is the processing engine used and evaluated in Chapter 5: Here, a modular *pipeline architecture* (Figure 2.10) is introduced. Complex analyses, comprising several basic processing tasks, can be performed on arbitrary large data sets, making optimal use of available hardware resources by parallelizing, interleaving, and distributing processing tasks alongside corresponding data subsets between computing resources (e.g., different servers in a distributed environment). Each analysis is described by a processing pipeline that defines involved processing tasks and can be reconfigured and replaced at runtime. Efficient retrieval of arbitrary subsets is achieved by means of a multi-layered hierarchical subdivision: For each point cloud, a separate spatial data structure is generated that best compliments the spatial distribution of the corresponding points (e.g., quadtrees for airborne data sets, octrees or kd-trees for terrestrial data sets). In turn, those spatial data structures are integrated into an overarching quadtree, allowing to efficiently answer queries stretching across multiple point clouds.

2.5.1 Pipeline Architecture

The pipeline architecture developed as part of this thesis comprises two major components: First, a *resource manager*, monitoring the memory and processing capacity of a system and distributing them among currently executed processing tasks (Section 2.5.2); second, a *pipeline engine* to configure and execute various pipelines plans, each of which defining a specific combination of basic input, processing and output tasks. To be more precise, a pipeline plan may contain the following elements (also referred to as *pipeline nodes*):

- **Importers**, i.e., pipeline nodes that import point clouds from any source such as files, a point cloud database or external sources (e.g., web services). For each

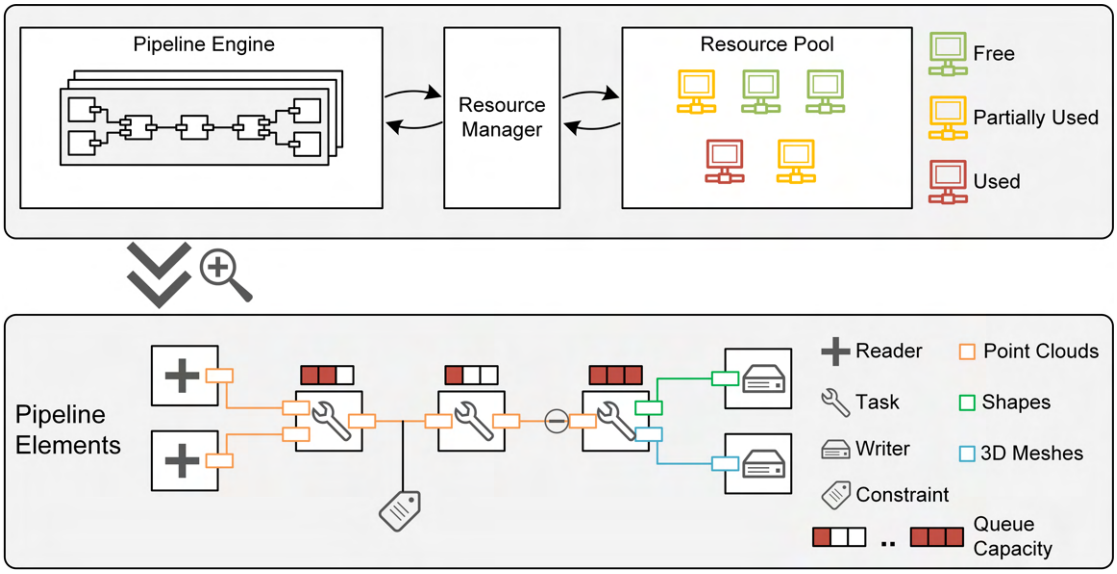


Figure 2.10: *Overview of the pipeline architecture and pipeline elements.*

data source and file format (e.g., LAS, E57) a separate importer is provided. Each importer prepares *data packages*. If the input data exceeds the maximum data package size, the importer prepares subsets by splitting the data.

- **Exporters**, i.e., pipeline nodes that export processing and analysis results into standardized formats for point clouds (e.g., LAS, E57), use-case specific LoD representations and point cloud databases, or other common geodata formats (e.g., shape files, CityGML, GeoTIFFs).

- **Tasks**, i.e., pipeline nodes that implement a specific processing or analysis algorithm. Some algorithms operate on multiple data packages simultaneously (e.g., to compare or to merge them). Similarly, algorithms may split incoming data sets or yield multiple results (e.g., additional per-point attributes and corresponding shapes). Hence, multiple incoming and outgoing connections may be defined per task.

- **Connections**, i.e., links between two pipeline nodes for the transfer of data packages. They define the order of execution. A given connection transfers only packages of a specific type (e.g., point clouds or shapes). Depending on the pipeline nodes being connected, various constraints may be defined, such as specific data layers that are required.

- **Data Packages**, i.e., data subsets that are transferred between pipeline nodes via *connections*. Like connections, a given data package may only contain a specific type of geodata. Also, the size of the corresponding data subset may not exceed a specific maximum defined by the resource manager.

External services for geodata can be seamlessly integrated by implementing tasks and

importers as interfaces. Vice versa, by adding exporters for specific external data formats, the proposed pipeline architecture may be easily integrated into existing workflows and systems. Pipeline plans are executed by the *pipeline engine*. Several pipeline plans can be executed in parallel; every single one can also be dynamically started, paused, and stopped. At runtime, each active pipeline node gets assigned its own set of resources by the *resource manager*, responsible for monitoring and distributing memory and processing usage within a system. Processed data packages are immediately transferred to subsequent pipeline nodes. Pipeline nodes manage a queue of incoming data packages for each incoming connection, whose size is restricted to a maximum number of data packages that can be defined at runtime. If a queue reaches its maximum capacity, no additional data packages are accepted and preceding nodes are not executed for the moment. To improve their runtime performance, the most time-consuming pipeline nodes are executed in parallel by adaptively assigning additional resources (e.g., CPU or GPU cores).

2.5.2 Memory and Resource Management

The resources of a system may be distributed across several network nodes, each featuring different memory capacities (i.e., size of secondary storage, main memory, and GPU memory) and computing capabilities (e.g., number and clock speed of CPU and GPU cores, memory transfer rates). Network nodes and their resources are added to a global *resource pool* that is monitored by the resource manager. Whenever a pipeline node needs to be executed, the resource manager assigns resources based on available memory and processing capabilities. After the execution is finished, all assigned resources are released to the resource pool and become available for other pipeline nodes (Figure 2.10). Distributing resources requires the resource manager to make a trade-off between several, often contradicting optimization goals:

- **Exclusivity.** Exclusive access to a resource (e.g., storage or GPU) significantly improves the runtime performance of a pipeline node (e.g., by minimizing cache misses and seek times).
- **Transfer Costs.** Frequently transferring data packages via connections may notably reduce the performance if subsequent pipeline nodes operate on different network nodes. This can be avoided by executing them on the same network node.
- **Parallelization.** Executing pipeline nodes in parallel or interleaved is an essential mechanism to improve the overall performance of the system. Thus, available resources and network nodes should be shared among as many pipeline nodes as possible.

The runtime of pipeline nodes may vary significantly depending on the corresponding operation. Thus, an adaptive resource scheduling allows to manage bottlenecks: The execution time is tracked for each pipeline node and the number of assigned resources is adjusted dynamically, whenever necessary.

2.6 Point Cloud Visualization Concepts

A general overview of point-based rendering is given by Gross and Pfister [71]. Several rendering techniques aim for a photorealistic and, thus, solid visualization of point clouds without holes in the surface [163, 194]. These techniques commonly represent points as splats, i.e., oriented flat disks [28, 196, 14], spheres, or particles. To visualize closed surfaces, an adequate size and orientation must be applied to each point [85]. These attributes can be calculated in a preprocessing step [188] or on a per-frame basis as proposed by [127]. However, these techniques are often difficult to apply because of varying point densities, e.g., on horizontal and vertical structures, as well as on fuzzy and planar areas. In addition, it is difficult to combine these techniques with out-of-core rendering techniques for point clouds because the point density varies depending on the LoD. As an alternative that scales better for massive data sets, visual artifacts can be eliminated via post-processing using image-based rendering techniques, e.g., to fill holes [52, 146], to blur visual clutter [96], or to emphasize depth cues [29, 104].

Non-photorealistic rendering techniques for point clouds have been proposed by Goesele *et al.* [67] and Xu *et al.* [191]. The silhouette highlighting technique of Xu et al was extended and added to the set of rendering techniques used in the context of this thesis, particularly Chapter 3. Olson *et al.* [110] show how the complete set of silhouette points of a surface can be calculated instantly. However, that information comes with the cost of an additional preprocessing step.

Out-of-core rendering systems for point clouds have been presented in detail by several authors [66, 186, 141, 69]. These systems use LoD data structures that aggregate or generalize points solely based on spatial attributes. This is not applicable for the overall purpose discussed in this thesis since points need to be separated according to their various data layers at any time during rendering to apply context-aware rendering techniques as well as to render only points with specific information, e.g., selected surface categories.

In the following subsections related work specific to Chapters 3, 4, and 5 is discussed.

2.6.1 Interactive Visualization of Enriched Point Clouds

Surface category information is commonly used to extract mesh-based 3D models [195], e.g., vegetation, building, or terrain models. However, surface categories are rarely used to enhance the visual quality of a point cloud directly - aside from adapting the colorization of the points. A more advanced rendering approach that does take surface categories into account was presented by Gao *et al.* [64]. They aim for a solid, hole-free visualization of airborne laser scans by resampling terrain segments and by applying a solid rendering style. The purpose of this approach is quite similar to the one presented in Chapter 3. However, the approach presented here supports a larger variety of rendering styles that may be applied to arbitrary surface categories at runtime.

Interactive occlusion management techniques for virtual 3D environments –conceptually similar to the ones applied to point cloud depictions in Chapter 3– have been widely

discussed for years. However, the focus typically lies on mesh-based 3D models instead of point-based rendering. Elmqvist and Tsigas [58] differentiate three primary purposes for these techniques: (1) the discovery of yet unknown, occluded objects, (2) the exploration of occluded objects whose position has been known beforehand, and (3) the exploration of an occluded object's spatial relationship to its environment. Since all these purposes are important when exploring enriched point clouds, Chapter 3 introduces techniques addressing all of them. Elmqvist and Tsigas [58] identify five distinct categories of occlusion management techniques: Tour planner based approaches [18, 37] calculate special camera paths through an environment, ensuring that every significant object is visible at least once while following these paths. Some approaches use multiple views of an environment in parallel. The different views are either rendered into separate viewports [77, 34] or combined into one by applying multi-perspective projections [118, 35]. Volumetric probes stay with a single view for an environment. Instead, they deform objects (e.g., by manipulating their proportions) within a certain area that is often defined by the user [182, 168].

The see-through lenses for enriched point clouds proposed in Chapter 3 belong to the category of *virtual x-rays*. These techniques leave object proportions intact and usually operate in screen space, either masking out occluders completely or making them (semi-)transparent. They can be either active, i.e., requiring user input to define the areas they should be applied to [173, 145], or passive, i.e., finding suitable areas automatically based on available information about the objects within a virtual environment [175, 164]. By choosing to adapt the concept of visual x-rays, users are able to freely navigate the point cloud (which wouldn't be the case for tour planner based approaches) without having to manipulate or deform the data (which would be the case for volumetric probes). Moreover, the idea to combine multiple views was rejected, as this would interfere with existing out-of-core rendering techniques for point clouds.

2.6.2 Immersive Visualization of Point Clouds using VR Technology

Regardless of the type of rendered geometry, real-time rendering is based on performance optimization techniques, that reduce and simplify the geometry needing to be processed by the different stages of the rendering pipeline [12]. However, while techniques such as view frustum culling and detail culling can be easily applied to point clouds, occlusion, backface, and portal culling are designed with mesh-based geometry and closed surfaces in mind. Due to the unstructured nature of point clouds those techniques require adaptation before being applicable to point-based rendering. The rendering system presented in Chapter 4 implements occlusion culling based on the reverse painter's algorithm [78]. Backface and portal culling were not adapted, as both techniques require specific knowledge or preprocessed additional data layers about a point cloud (e.g., per-point normals, surface categories) that might not always be available. Performance optimization techniques specifically for VR applications have been discussed by Vlachos [177]. Some of those techniques, such as the hidden mesh or the single-pass stereo rendering, are implemented and evaluated by the rendering approach presented in Chapter 4.

Visual optimization techniques for point clouds primarily encompasses reduction of visual clutter and holes between neighboring points. Corresponding photorealistic rendering techniques for point clouds have already been discussed at the beginning of this chapter. However, several of these techniques introduce a noticeable performance hit and are thus not usable in the context of VR applications. Specifically for VR applications Schütz [160] introduces instead the usage of point cloud mipmaps as well as multisampling for a reduction of z-fighting and softer edges, which is evaluated in detail in Chapter 4.

Locomotion and interaction in virtual environments has been discussed by several authors. Studies from Wloka and Greenfield [187] and Sarupuri *et al.* [155] have shown that *six-degrees-of-freedom* (6DOF) input devices are preferred over more traditional devices such as keyboards, gamepads, or haptic gloves. Objects that can be both seen and touched reinforced the user's sense of presence while locomotion based on the 6DOF input device was much less likely to induce nausea. *Walking-in-Place* (WIP) as a technique for locomotion was introduced by Slater *et al.* [165]. User studies comparing WIP to joystick flying and real walking [166, 174] showed that techniques resembling walking in the real world gave the participants a stronger feeling of presence and less discomfort than artificial locomotion. To utilize this finding and cope with the challenge of virtual space being indefinitely bigger than physical space, *Redirected Walking* (RDW), originally proposed by Razzaque *et al.* [131], alters the user's path by slightly rotating the virtual world to keep the user within the available physical space. Studies [132, 130] have demonstrated that RDW can be effective while being unnoticeable by participants. This concept has been augmented in the past years. Chen *et al.* [41] proposed an algorithm to redirect the user through irregularly shaped environments with dynamic obstacles while Sun *et al.* [170] demonstrated that saccadic suppression and the subsequent temporary blindness can be used to increase the rotation gains without the user noticing. The *Point & Teleport* (P&T) locomotion technique was proposed and evaluated by Bozgeyikli *et al.* [32]. In an experiment with WIP and joystick flying, they found P&T an intuitive, easy to use, and fun technique, though not more immersive or less prone to induce nausea as the other two. Still, the participants rated it as their preferred technique.

2.6.3 Web-Based Rendering of Enriched Point Clouds

Point-based rendering approaches that combine out-of-core and web-based rendering concepts to enable the visualization of point clouds across device ecosystems [143] have become increasingly popular in recent years.

With *Potree*, Schütz and Wimmer [159] propose a thick client approach for arbitrary large data sets, which is adapted by Martinez-Rubi *et al.* [98] to interactively present a massive data set of the Netherlands. An alternative thick client renderer for point clouds named *Plasio* was introduced by Butler *et al.* [38]: Using open-source libraries such as Entwine and Greyhound massive data sets can be streamed interactively. *GVLiDAR* [50] and *ViLMA* [49] constitute thick client rendering approaches that focus on geospatial analysis and measurement tools. While all four frameworks provide effective interaction

and inspection techniques specifically for point clouds, they offer only minimal support to integrate additional, context-providing geodata (e.g., shapes, 2D maps). The *Cesium* [1] framework on the other hand aims to provide a generalized thick client rendering solution for arbitrary types of geodata (e.g., point clouds, 3D meshes, 2D maps). The approach presented in Chapter 5 is in parts based on that framework, expanding it by several context-aware rendering techniques and a set of interaction techniques for the collaborative inspection of enriched point clouds (e.g., to share, query, and annotate).

In addition, the presented approach also provides a thin client renderer, allowing to optionally reduce the performance impact on client-side by delegating the rendering to the server side. Compared to the aforementioned frameworks, it can thus adapt to a broader range of computing and graphics capabilities on client side. Similar approaches have been successfully implemented in the past [75, 43, 55] but typically focus on mesh-based geometry rather than point clouds. To generate stereoscopic panoramas the theoretical concepts described by Peleg *et al.* [121] were implemented by means of modern 3D computer graphics.

Systems for the efficient management of enriched point clouds have been recently presented and evaluated by several publications [44, 112, 124]. However, those contributions focus on the efficient storage, retrieval, and processing of the stored data sets. Less emphasis is put on the collaborative exploration, inspection and manipulation of the stored data sets.

[1] `https://cesiumjs.org`

Chapter 3

Interactive Visualization of Enriched Point Clouds

In this chapter, rendering and interaction techniques are presented and evaluated that take advantage of additional data layers (e.g., surface categories), thus facilitating the exploration and inspection of enriched point clouds, even for large-scale data sets. The chapter is based in parts on the author's scientific publications in [5] and [7].

The following sections are structured as follows: Section 3.1 further motivates this chapter and gives a more detailed introduction. Point-based rendering techniques that incorporate additional data layers as well as interactive and view-dependent see-through lenses are presented in Section 3.2, highlighting their specific advantages and disadvantages in different scenarios. Section 3.3 focuses on the implementation, introducing (1) a multi-pass rendering approach allowing to dynamically combine and configure these different rendering techniques as well as (2) an out-of-core rendering approach based on a layered, multi-resolution kd-tree. As shown in Section 3.4, this allows to provide interactive frame rates. Section 3.5 gives conclusions and outlines future research directions.

3.1 Introduction

The interactive exploration, inspection, and presentation of enriched point clouds is an essential functionality for an ever-growing number of geospatial and non-geospatial applications, as it enables their visual inspection, facilitates their interpretation, and provides an interface to perform analyses. While standard rendering systems for point clouds enable an interactive exploration of arbitrary large data sets by using fitting spatial data structures and LoD concepts [66, 186, 141, 69], these systems generally render all points in a uniform way, which complicates the visual identification and categorization of objects and structures by the user: For example, if points are rendered by the point primitives of the underlying rendering system (e.g., OpenGL's GL_POINTS) they are not scaled according to the camera distance, thus, making it difficult to correctly estimate depth differences and leading to visual artifacts due to overlapping of points close to each other. Furthermore, point clouds may exhibit stark differences in local point density by nature of the capturing process, i.e., based on the distance of the captured surface to the scanning device, the scanning angle, and possibly occlusions. Sometimes, reduced local point densities can even be pinpointed to specific surface categories, such as building

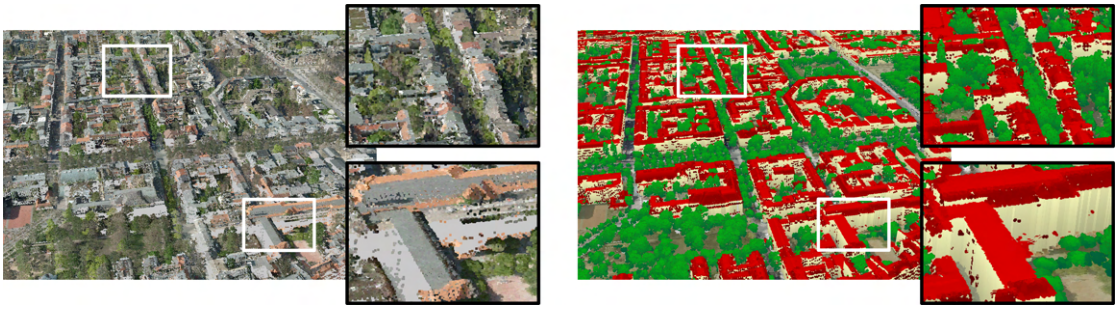


Figure 3.1: *Left: Example of a point cloud rendered in a uniform way by GL_POINTS primitives and textured by aerial photography. Right: Same scene rendered by a thematically enriched rendering technique: Different surface categories can be better distinguished, holes on façades are filled, and visual clutter in the background is reduced.*

facades or water surfaces within airborne laser scans. Standard rendering, therefore, leads to gaps between neighboring points in some areas, which complicates their perception as a continuous surface (Figure 3.1).

In more general terms, common rendering systems do not include additional data layers into their rendering strategy as a criterion, and therefore tend to give an equal amount of significance to every point. However, the relevance of a point in the context of spatial information often depends on additional data layers (e.g., timestamp, surface category, RGB photographic color, spatial position) as well as the current visualization or exploration task. In this chapter, we focus on the following use cases:

- **Point Clouds from Different Acquisition Techniques.** Spatial data for a site may result from different surveys such as an airborne, mobile mapping, terrestrial, outdoor and indoor data acquisition. On a broader scale, point clouds can consist of overground as well as subterranean structures (e.g., mine shafts, sewers). By allowing users to see through occluding structures (i.e., by masking out corresponding points in front of the others) an in-depth exploration of such point clouds can be facilitated (Figure 3.2).

- **4D Point Clouds.** Many applications include repetitive scans and simultaneous use of point clouds captured at different points in time. To assist users in exploring spatial differences such as structural changes of the captured site over time (e.g., constructed, demolished, or modified buildings), points indicating such changes should be highlighted (Figure 3.3).

- **Classification-dependent Rendering.** Typically, points represent different surface categories (e.g., ground, building, vegetation, water, city furniture). Standard rendering does not take into account characteristics of these surface categories (e.g., fuzziness of vegetation, smooth ground surfaces or planar building roofs). For that reason, it limits the visual perception of such categories as well as the identification of individual objects (e.g., individual trees and buildings) and relations between

Figure 3.2: *Example of a point cloud consisting of indoor and outdoor scans. It is explored with a see-through lens to inspect the occluded interior of the building in the context of the overall scan.*

them. Furthermore, relevant objects might not be visible at all due to occlusion by less relevant objects (e.g., buildings below vegetation), necessitating rendering techniques to mask out less relevant points in such occlusion events (Figure 3.4).

The visualization of enriched point clouds can be improved by taking into account additional data layers that further characterize the surface represented by a point. These data layers can be derived from external data sources (e.g., aerial images, thematic maps) or computed by point cloud analysis approaches [94, 39, 139] that typically analyze geometric relationships between points such as connectivity, local flatness, normal distribution, and orientation. In the following sections, a novel rendering approach is presented that uses available data layers, such as timestamps, surface categories, thematic and geometric information, to adapt the appearance of each point, i.e., its color, size, orientation, and shape. Different photorealistic, non-photorealistic, and solid point-based rendering techniques matching different surface characteristics are selected on a per-point basis. The different rendering techniques can be configured at runtime according to the visualization or exploration task. To filter points based on their relevance to the given task, see-through lenses for enriched point clouds are defined as follows:

- A see-through lens defines a space within a point cloud, in which points of higher relevance are emphasized by masking out less relevant points completely or in parts.
- The area of a see-through lens can be defined interactively by the user or automatically with respect to the current view position (e.g., center of the screen), i.e., view-dependent.

Figure 3.3: *4D point cloud enriched with surface category information: Points representing buildings that are not present in the earlier scan (upper left) but captured in the new scan (lower right) are highlighted with a red color scheme and by tracing their boundaries.*

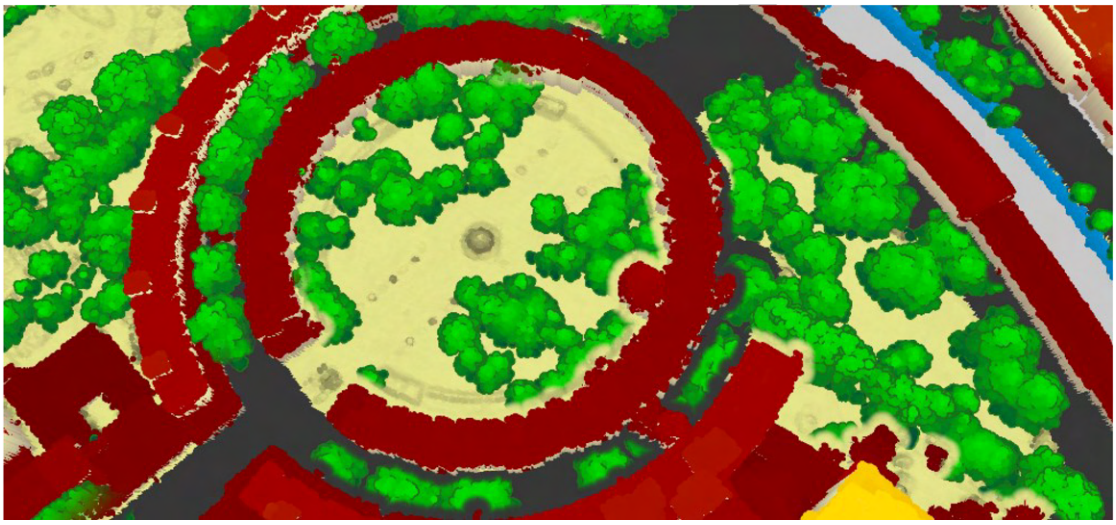


Figure 3.4: *Enriched point cloud with surface category information: Points representing vegetation are masked out in the lower right part to show hidden building parts.*

In this chapter several interactive and view-dependent see-through lenses for enriched point clouds are introduced. Interactive visualization of point clouds, that exceed available memory resources and rendering capabilities, is achieved by storing points in a layered, multi-resolution kd-tree providing a surface category specific subdivision of the data.

3.2 Visualization Concepts for Enriched Point Clouds

In the context of this chapter, we focus on enriched point clouds and 4D point clouds containing the following data layers:

- **Geometric Information**: *Surface normals* approximate the surface of the local point proximity. They can be computed efficiently by analyzing the local neighborhood of a point [103] and are used to orientate the point primitive according to the represented surface.

  The *horizontality* indicates how vertical the surface normal of a point is oriented, i.e., points representing horizontal surfaces (e.g., flat building roofs) feature higher values than points on vertical surfaces (e.g., building façades) [195]. It can be used for a colorization to accentuate detailed and geometrically complex object structures (e.g., roof elements).

  The *local height* describes the height of a point in relation to all points belonging to the same surface category in the point's proximity. Using local heights for a colorization allows to highlight edges and differences in the structure of an object, e.g., roof ridges and smokestacks.

  The *global height* describes the height of a point in relation to all other points that belong to the same surface category. Colorizing points based on their global height emphasizes height differences for different objects belonging to the same surface category (e.g., trees with different heights).

- **Graphical Information**: Color or color-infrared values can be extracted from aerial images, ideally captured at the same point in time as the 3D point cloud. These values are generally used for a colorization, e.g., when a photorealistic and natural appearance of the points is required.

- **Surface Category and Object Information**: A basic categorization of points into indoor, outdoor, overground, subterranean, airborne, or terrestrial can be made directly during the capturing process. A more specific categorization into surface categories such as building, ground, and vegetation can be derived with point cloud classification approaches that are based on analyzing topological attributes or additional geodata capturing the same surface area (e.g., infrastructure maps) [139].

- **Thematic Information**: *Time stamps* describe the date of data collection.

  Change information describes on a per-point basis the degree of change related to a given reference geometry. That reference geometry might be another 3D point

cloud captured at a different point in time or a 3D mesh representing the same surface area, e.g., building model, cadastre data, 3D city model.

While the rendering techniques presented in this chapter have been implemented with these specific data layers in mind, they may also be used analogously in combination with other data layers, as long as those are available on a per-point basis.

3.2.1 Point-Based Rendering Techniques

All aforementioned data layers can be used to adapt the appearance of a point, i.e., its color, size, orientation and shape, at runtime. The color of a point can be chosen based on its color attributes, surface category, topologic information (i.e., surface normal, horizontality, global, or local height), temporal or change information, or a combination of these. The orientation of a point can either correspond to its surface normal, the current view direction or a defined uniform vector. In addition, size and shape type of a point can be set dependent on its surface category. Regarding the shape type, different rendering techniques may be applied, each coming with different strengths and weaknesses:

Default Point Primitives

To efficiently render 3D point clouds, the *Graphics Processing Unit* (GPU) supports point primitives, such as *GL_POINTS* in OpenGL. However, these primitives have a fixed size in pixels (e.g., Figure 3.5 (a) uses a size of three pixel), i.e., their size in object space varies according to their perspective depth. Depending on the view position, undersampling, i.e., holes between neighboring points (Figure 3.5 (a) - bottom), or oversampling, i.e., visual clutter due to overlapping points (Figure 3.5 (a) - top), occurs.

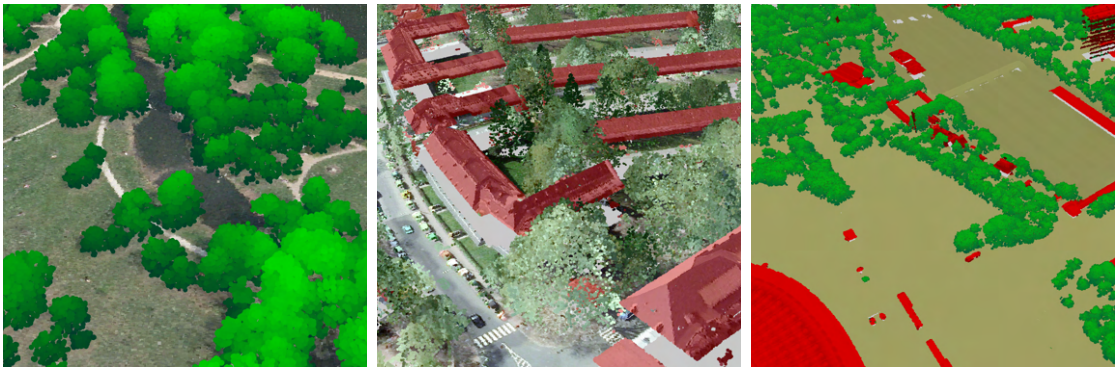
Point Splats

To avoid undersampling and oversampling due to changing view positions, the point splats technique renders each point as an opaque disk defined in object space that can be oriented alongside the surface normal [150, 28]. The on-screen size depends on the current view position and angle, ensuring a perspective correct visualization (Figure 3.5 (a–f, i)). However, the perception of depth differences between overlapping points that are colored homogeneously (e.g., points belonging to the same surface category), is generally limited.

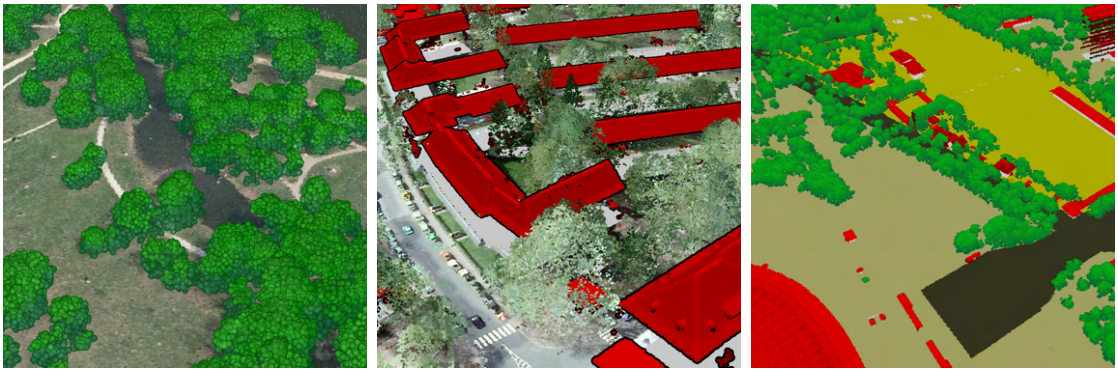
Point Spheres

This point-based rendering technique emphasizes the three-dimensional character of a point. The proposed point spheres extend the original splat concept by rendering points as hemispheres instead of flat disks that are always facing the view position and, thus, look like spheres [150]. These hemispheres are created by (1) adding an offset to each depth value of the rendered fragment and by (2) shading each fragment. The depth offset as well as the shading color can be determined by projecting the fragment onto a plane defined by the corresponding splat and by calculating the projected distance of

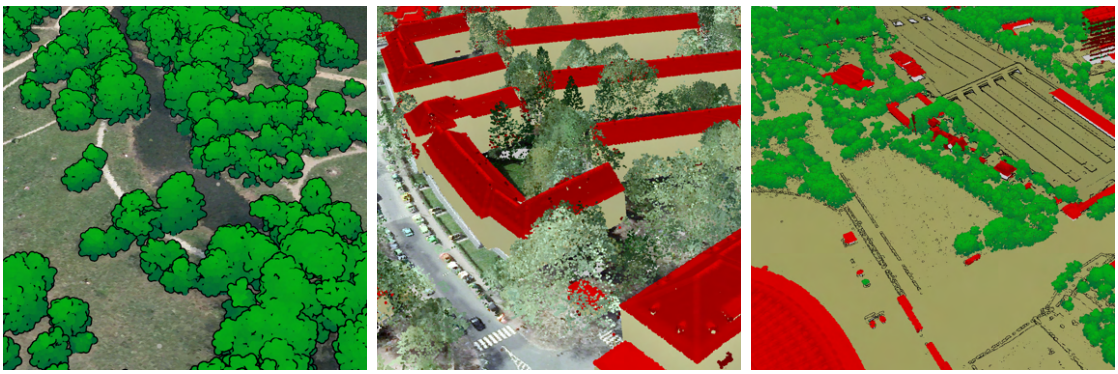
(a) *Point Splats; aerial image colors.* (b) *Point Splats; aerial image colors.* (c) *Point Splats; aerial image colors.*



(d) *Points Splats; global height.* (e) *Point Splats; aerial image colors and surface category information.* (f) *Point Splats; global height.*



(g) *Point Spheres; local height.* (h) *Silhouette Rendering; horizontality.* (i) *Point Splats; surface category information.*



(j) *Silhouette Rendering; local height.* (k) *Solid Rendering; horizontality.* (l) *Silhouette Rendering; global height.*

Figure 3.5: *Examples of enriched point clouds rendered with different rendering setups for different surface categories: Vegetation (left), buildings (middle), and terrain (right).*

the fragment to the center of the splat. Point spheres are well suited for non-planar and fuzzy surfaces, such as vegetation (Figure 3.5 (g)).

Silhouette Rendering

Point-based silhouettes highlight and abstract silhouettes and distinctive surface structures (e.g., depth differences). This technique extends the splat rendering approach and was originally proposed by [191]. Similar to the rendering of point spheres, color and depth of each fragment depends on its projected distance to the center of the splat. In addition, the splat is divided into an inner and an outer part. Fragments in the outer part represent the silhouette and are rendered with an increased depth value and a distinct color. As a result, depth discontinuities between overlapping points exceeding a given depth offset are highlighted (Figure 3.5 (h, j, l)).

Solid Rendering

In this context, solid rendering of point clouds focuses on building façades (or similar, primarily vertical surfaces), aiming for a solid and hole-free depiction. As the point density on façades in airborne laser scans is very low in contrast to horizontal structures, the efficient identification of building segments is limited because other structures behind a building are visible through the façade [64]. To overcome this, a second rendering pass is used to fill the area below roof points with additional primitives. The *geometry shader* is used to render (1) a point-based splat, sphere, or silhouette equal to the rendering techniques presented above and (2) a quad that imitates the façade below a point. The quad width is equal to the point size used in (1) whereas the height depends on the point's distance to the terrain level. All quads are aligned to the view direction and have the same color or height-based color gradient to create the appearance of a solid façade (Figure 3.5 (k)).

3.2.2 View-Dependent and Interactive See-Through Lenses

Adapting the appearance of points based on their data layers facilitates the visual identification of distinct objects within point cloud depictions by highlighting neighboring points that are related to each other (e.g., by representing the same surface category or by sharing the same time stamp). However, if points are occluding each other, the aforementioned rendering techniques alone are not sufficient to reliably highlight all distinct objects within a data set. Instead, these rendering techniques need to be combined with visual filtering and highlighting techniques that explicitly reduce visual clutter by masking out points based on their significance to the current application, use-case, and kind of information that needs to be explored. To describe the significance of a point the following priority levels are used:

- **Focus**. Essential information of interest and exploration aim (e.g., interior objects occluded by walls, subterranean structures or buildings that have been demolished

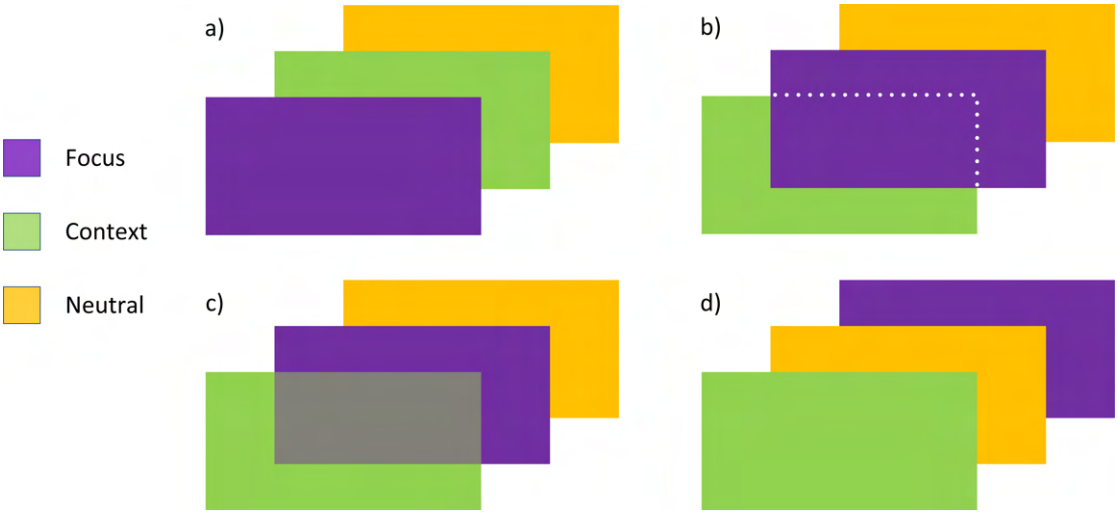


Figure 3.6: *Illustration of compositions for different priority levels in occlusion situations. Context information is (b) masked out in favor of or (c) blended with focus information. Neutral information is neither highlighted nor filtered.*

between consecutive scans). The more points carrying such information are occluded, the less information can be gathered, i.e., the less effective the exploration becomes.

- **Context**. Information that increases the overall realism of the visualization without being the main focus of the exploration (e.g., vegetation in densely forested areas). Points carrying such information can be safely masked out in favor of focus information.

- **Neutral**. Depending on the use case, users might want to focus on specific occlusion scenarios, such as solely on buildings that are occluded by vegetation. This requires to define all other information as neutral, i.e., points representing such information are treated as solid geometry that is neither masked out or highlighted.

These priority levels are used to define the composition of the data for the rendering and visualization of different occlusion scenarios (Figure 3.6): Occlusions with the same priority level or including points carrying neutral information are solved by displaying the nearest point to the view position (i.e., similar to the default behavior in 3D rendering). Points carrying context information on the other hand (so-called context points) are masked if focus points are occluded. To mask context points, different rendering techniques may be applied, each coming with different strengths and weaknesses:

Simple Blending

Simply masking out all context points in occlusion scenarios limits the correct estimation of depth differences and does not provide information about the object shape and boundaries within a point cloud depiction (Figure 3.7 (a)). Blending context points and

focus points by a certain factor addresses these limitations, however, areas with blended structures might be difficult to recognize during the exploration (Figure 3.7 (b)).

Blueprints

Blueprints are a traditional form of technical drawings and known for their characteristic style which originating from the historical contact print process [107]. Construction elements are visualized by tracing outlines using different line widths and a color contrasting favorably with the background. Thus, the focus of the viewer is directed towards the most significant construction elements. This concept can be adapted to highlight areas where focus and context points have been merged by tracing the boundaries of those areas with a configurable line width and color (Figure 3.7 (c)). This approach is especially effective to highlight changes within 4D point clouds (Figure 3.3).

Halos

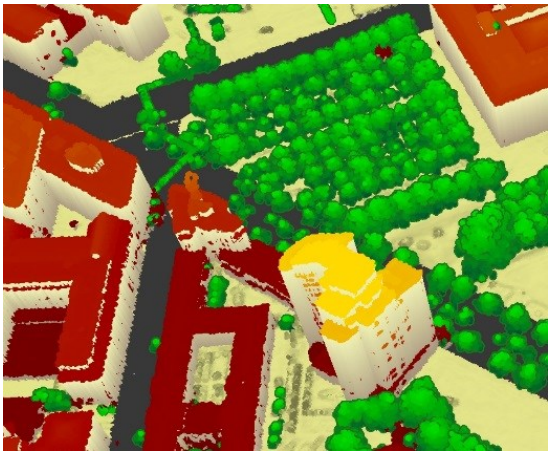
A stronger emphasis on focus points can be obtained by masking out additional context points in a defined local proximity (Figure 3.7 (d)). As a result, groups of neighboring focus points are surrounded with a halo effect, similarly to the real-world phenomenon and the technique used by artists throughout history to emphasize certain individuals. Techniques that highlight objects by removing occluders are known as cut-away-views [175, 164]. So far, they have not been applied to point-based rendering. The typical use case to apply this technique is the exploration of complex structures that are completely occluded by their surroundings, such as subterranean structures (Figure 3.2).

Interactive See-Through Lenses

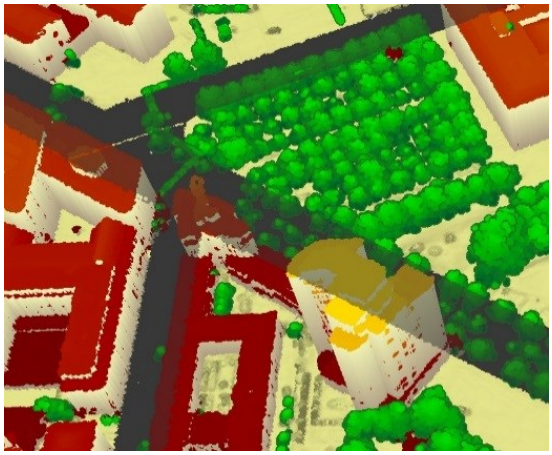
All techniques that have been introduced are applied automatically to the data where focus information is occluded by context information. Naturally, the number, position, and extent of these areas varies depending on the view position. As opposed to that automated, view dependent approach, interactive see-through lenses (also commonly known as 'magic lenses') can be moved freely across the screen. Within an interactive see-through lens, context points are masked out completely, whereas in the surrounding no blending is applied at all (Figure 3.7 (e+f)). This is required to focus on occlusions within certain areas whose position is known beforehand (e.g., to explore and show a former state for a certain area). However, if those areas are unknown, interactive see-through lenses are inefficient as the entire point cloud has to be traversed manually to identify all areas.

3.3 Out-of-Core Rendering and Image Compositing

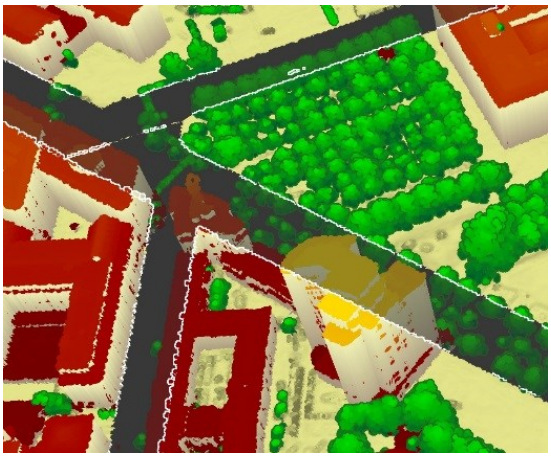
The interactive visualization of point clouds exceeding available memory resources and rendering capabilities demands for out-of-core rendering techniques that combine LoD concepts, spatial data structures, and external memory algorithms. In the following, a



(a) *No see-through technique applied.*



(b) *Simple Blending highlighting streets in favor of buildings and vegetation.*



(c) *Blueprints highlighting streets in favor of buildings and vegetation.*

(d) *Halos highlighting buildings in favor of vegetation.*

(e) *Interactive See-Through Lenses highlighting buildings in favor of vegetation.*

(f) *Interactive See-Through Lenses highlighting buildings in favor of vegetation*

Figure 3.7: *Examples of different see-through lenses applied to enriched point clouds.*

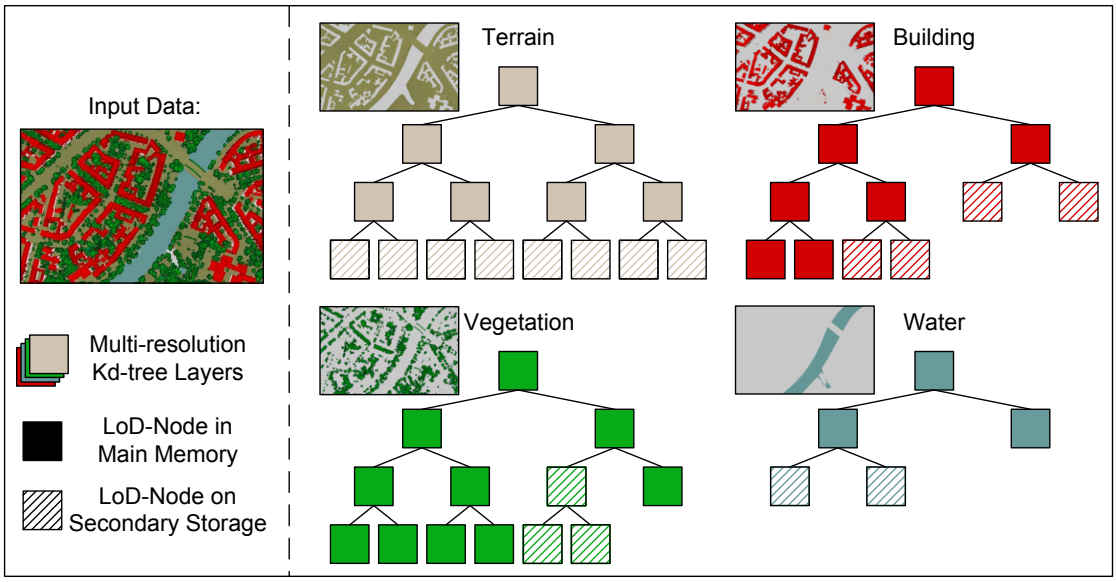


Figure 3.8: *Schematic overview showing the structure of our layered, multi-resolution kd-tree. For each surface category a separate multi-resolution kd-tree is maintained.*

layered, multi-resolution kd-tree for enriched point clouds is presented. It is characterized by the following properties:

- Adaptive multi-resolution LoDs to preserve a defined rendering budget (e.g., 30 frames per second).
- Efficient and adaptive memory management (e.g., by using equal-sized LoD chunks).
- Semantics based subdivision of the data to enable a selective access and visualization (e.g., only building points or only points captured at a specific point in time).
- LoD selection that that takes into account semantics to fulfill different requirements for specific rendering techniques (e.g., varying point densities).

While the concrete implementation described in this chapter subdivides the data based on the available surface categories, that subdivision may analogously be based on any other data layer that defines a discrete number of possible values (e.g., acquisition types).

3.3.1 Layered Multi-Resolution Kd-Tree

As described in Section 2.4, a variety of different spatial data structures is commonly used to organize point clouds. The construction of quadtrees and octrees can be performed faster in contrast to kd-trees because there is no need to sort the points. However, the use of quadtrees and octrees for irregular and sparse distributed data, e.g., airborne laser scans, results in tree nodes with a varying number of points. Out-of-core memory

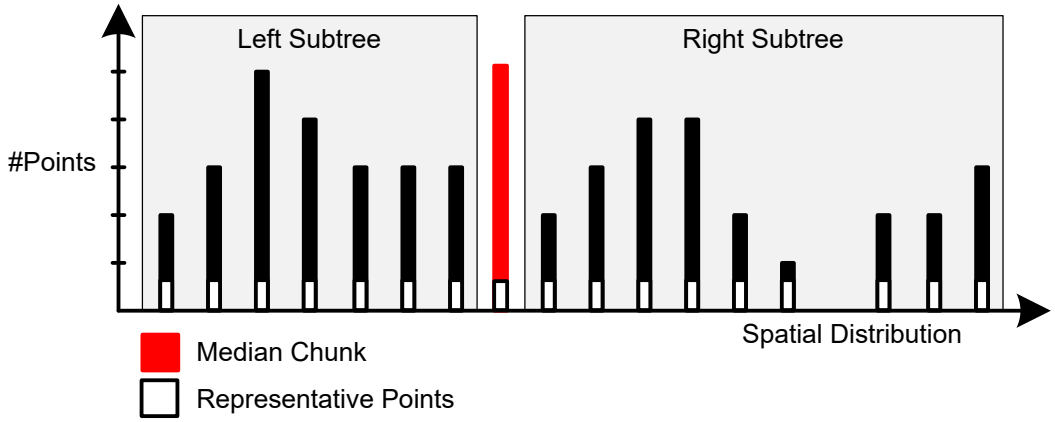


Figure 3.9: *Illustration of the histogram-based construction of the kd-tree to reduce preprocessing times.*

management must implement efficient caching and memory swapping mechanisms that benefit from *equal-sized* data chunks. For that reason, kd-trees were chosen to arrange the data in this instance. All points belonging to the same surface category are arranged in a sub-tree consisting of nodes with an equal number of points (Figure 3.8). Each of these nodes corresponds to a specific LoD for a spatial area with the root node representing the overall expansion of the point cloud and child nodes subdividing the area of their parent node. Each point is stored only once in the tree, and all nodes together are equal to the input point cloud.

The layered, multi-resolution kd-tree is constructed in a preprocessing step. It can be stored on secondary storage and therefore applied for arbitrary sized point clouds. First, the given point cloud is subdivided based on surface categories. Second, for each surface category the corresponding points are arranged in a multi-resolution kd-tree. The construction of a kd-tree with an equal number of points per node, i.e., a *balanced kd-tree*, is implemented by a multi-pass histogram-based approach that avoids a time-consuming sorting of the entire data for each tree level. In a first pass, we iterate over the point cloud to fill a histogram that describes the spatial distribution and extent of the data. Like a uniform grid, the histogram organizes points into a number of equal-sized spatial chunks. For each chunk, the number of points belonging to the respective area and a representative point are stored (Figure 3.9). Based on the number of points per chunk and the spatial extent of the histogram, a median chunk can be determined that contains the median point required to construct the kd-tree. A second iteration over the point cloud is used to fill up the current node with representative points (i.e., to create a LoD) and to assign all points to the left or right part of the tree. Only points belonging to the median chunk need to be sorted to determine the exact median element. The median element for the split is chosen so that the number of points to the left is a multiple of the

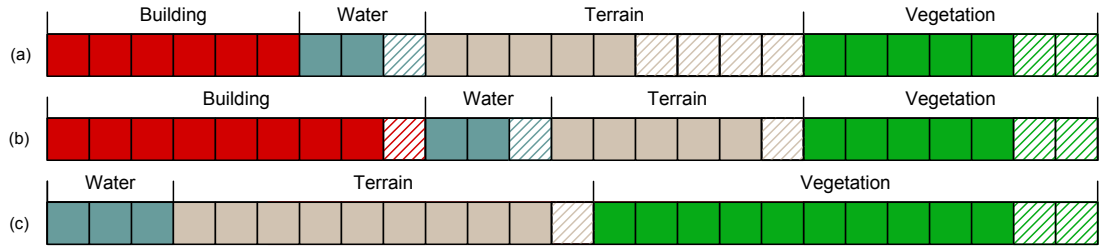


Figure 3.10: *Illustration of an exemplary GPU memory usage that is balanced during rendering according to memory requirements of LoD nodes that belong to different surface categories. (a) to (b) illustrate how unused memory is assigned to other surface categories. (b) to (c) illustrate the balancing process when the visualization of one surface category (e.g., building) is disabled.*

number of the points stored per node. This is important to construct a balanced kd-tree with equal sized nodes with exception of one leaf node. The out-of-core construction process subdivides point data on the file system until data chunks can be processed in main memory.

3.3.2 Layered Kd-Tree Rendering

The rendering process can be divided into three stages that are performed per frame. The first stage is responsible for the data provision, caching, and transferring of points from secondary storage to main memory as well as from main memory to GPU memory using the layered, multi-resolution kd-tree. The second stage applies one of the point-based rendering techniques described in Section 3.2.1 to all points belonging to the respective surface category. The final stage seamlessly combines all surface category specific rendering results into one final image (Section 3.3.3).

At first, the root nodes of all surface category specific sub-trees are loaded into main memory. Each chunk is equal to a LoD node and is mapped into a *vertex buffer object* (VBO) resident in GPU memory. The VBO is divided into equal sized chunks that can store exactly one LoD node. The layered, multi-resolution kd-tree is used to determined LoD nodes that need to be transferred to or can be removed from the VBO. The decision to add or remove a LoD node from memory depends on the *projected node size* (PNS). Therefore, the bounding sphere of the node is projected into screen space, and the number of covered pixels is compared to the number of points per node [141]. The threshold applied to the PNS depends on the point-based rendering technique, available memory, and computing capability of the GPU. Each surface category has its own memory budget (Figure 3.10) and is balanced permanently during the rendering process because the amount of memory required by a surface category may vary due to the following reasons:

- Only a small number of points belonging to a surface category is visible during the exploration.
- Visualization of certain surface categories is disabled.

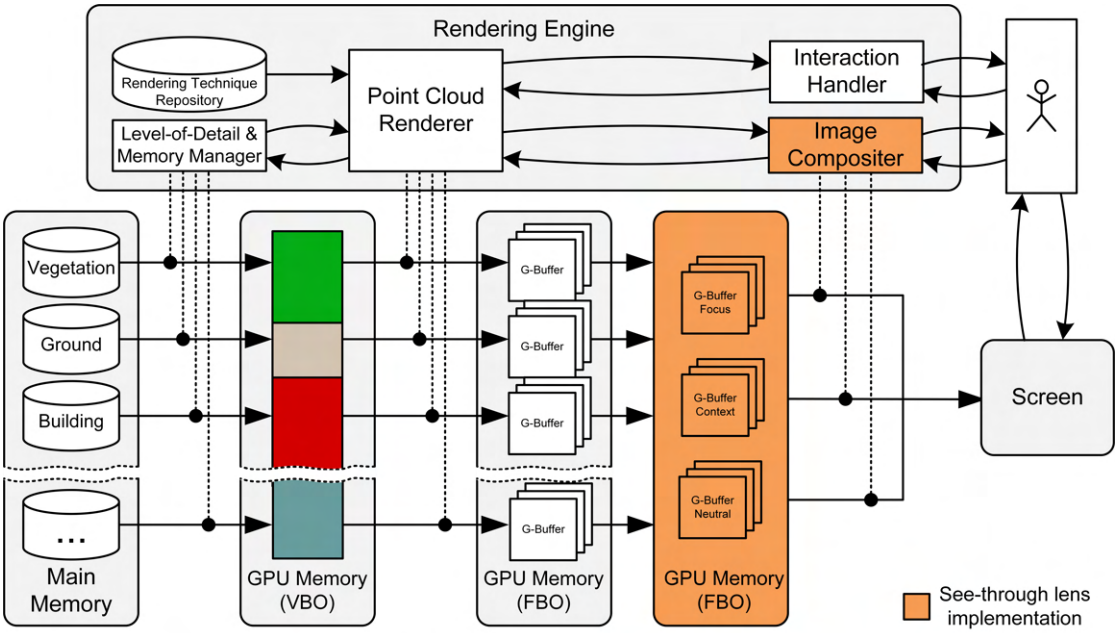


Figure 3.11: *Schematic overview of the rendering system described and evaluated in this chapter. Categorized by surface categories, points are transferred to GPU memory and rendered into separate G-Buffers that are merged based on the respective priority levels before being composed to the final image.*

- Close up views require a high point density for a surface category (e.g., for buildings).

Surface categories can be rendered with different LoDs because the required number of points for an appropriate rendering result depends on the structure. For example, buildings may require to be rendered with more points due to detailed roof structures in contrast to terrain or vegetation that can be rendered with less points. To ensure a hole-free surface, the lower point density can be compensated by using larger primitives, e.g., splats for terrain or spheres for vegetation.

3.3.3 Image Compositing

We combine different point-based rendering techniques by a multi-pass rendering techniques based on so-called G-Buffers[152] for image-based composition. (Figure 3.11). *G-buffers*, from a rendering perspective, are specialized *frame buffer objects* (FBO) that store multiple 2D textures such as for color, depth, or normal values. In our approach, we use a separate rendering pass for each surface category; the results of each rendering pass are stored in a corresponding G-Buffer. The G-Buffers are then merged based on the priority levels assigned to the respective surface categories. To synthesize the final image, the G-Buffers are drawn onto the canvas. This compositing pass allows us, for example, to implement see-through lenses by programmable fragment shaders and, therefore, to activate or deactivate the lenses at run-time.

In our implementation, the compositing pass always combines exactly a fixed number of G-Buffers, independently from the overall number of surface categories within an enriched point cloud. For our use cases, three G-Buffers turned out to be sufficient for the current functionality of the rendering techniques and for the variety of additional layers, in particular, surface categories.

To enable halos with a defined proximity (e.g., two meters in world-space) and additional halo mask must be prepared for surface belonging to the focus that determines the area in which context information needs to be maxed out. However, creating that mask for each frame has a notable impact on the overall performance (Section 3.4).

G-Buffer-based Interaction Techniques

Using G-Buffers also provides the technical foundation for interaction techniques that allow for an in-depth inspection of enriched point clouds: Upon generating the layered, multi-resolution kd-tree, every single point gets assigned a unique identifier, detailing (1) its corresponding sub-tree, (2) the node within that sub-tree, and (3) its exact position within the point set of that node. Similar to other data layers, these point-specific ID values are first rendered into a separate sub-texture of the corresponding G-Buffer. The subsequent compositing passes then merge all ID sub-textures into a single entity that can be accessed at runtime, allowing to trace any fragment of the final image back to the corresponding point and all its data layers with minimal effort.

Based on this mechanic, a variety of interaction techniques can be implemented, such as:

- **Live Picking.** While hovering with the mouse cursor over the point cloud depiction, any available information about the point underneath the current mouse cursor position (i.e., its exact coordinates as well as any other semantics) is displayed.

- **Live Measuring.** Points underneath the current mouse cursor position may be selected via double clicking to interactively define line segments, areas, and volumes, the size of which is calculated and displayed on the fly.

- **Object Selection.** Contingent on the processing steps and analyses conducted prior to the visualization, data layers may be available that define ID values to assign each point to a specific object (e.g., a specific tree or building segment) amongst the corresponding surface category. In that case, double clicking selects and highlights the complete object the point underneath the current mouse cursor position belongs to, rather than only the point itself. Once selected, available information (e.g., number of points, area or volume size) about an object is displayed.

These basic interaction techniques can be easily combined and extended into more sophisticated ones (e.g., GIS-like features, a camera-path editor), thus, allowing to design and implement applications that are highly customized to specific use cases.

Table 3.1: *Characteristics of the data sets used to evaluate the performance of the presented rendering approach.*

	Data Set 1	Data Set 2	Data Set 3
Point Density	10 pts/m^2	28 pts/m^2	100 pts/m^2
Number Points	5 Billion	7.1 Billion	80 Billion
Data Size	112 GB	159 GB	1788 GB

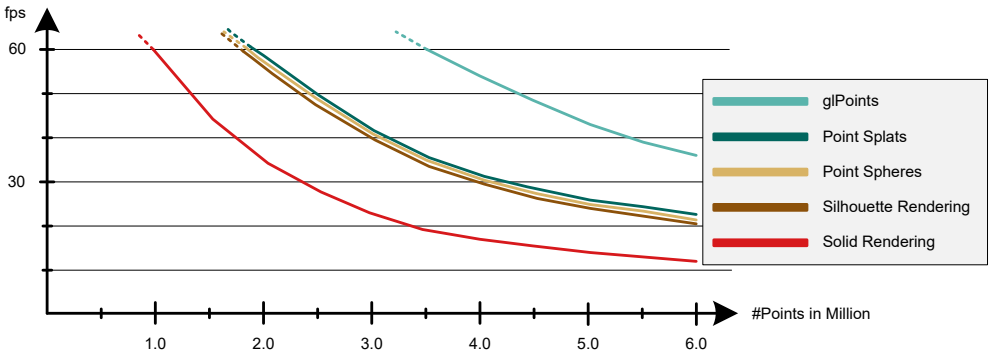


Figure 3.12: *Rendering performance in frames per second (fps) using different sized subsets of the data sets from Table 3.1.*

3.4 Performance Evaluation and Results

The presented rendering approach was implemented based on C++, OpenGL, GLSL, and OpenSceneGraph. Performance evaluation was conducted based on three real-world data sets of different size and point density containing up to 80 billion points (Table 3.1). The test system consisted of an Intel Xeon CPU with 3.20 GHz, 12 GB main memory, and a NVIDIA GeForce GTX 770 with 2 GB device memory.

Each data set was rendered from a zoomed out and a close-up perspective which affected the number of points rendered for a given frame. As depicted in Figure 3.12, interactive frame rates can be achieved for each point-based rendering technique as long as the overall number of rendered points does not exceed a certain threshold (e.g., six million points for the solid rendering approach). The highest frame rate could be observed for *GL_POINTS*, which was expected since these primitives are supported natively by the GPU. Point Spheres as well as the solid and the silhouette rendering approach extend the concept of Point Splats and increase the computational effort during rendering. Consequently, lower frame rates were achieved when using these techniques for rendering as opposed to Point Splats. Furthermore, the observed performance for Point Spheres is higher than for Point Silhouettes due to a more hardware demanding shading implementation (e.g., conditional branching).

With respect to see-through lenses, applying simple blending, blueprints, and interactive see-through lenses has minimal effect on the overall performance (Tables 3.2

Table 3.2: *Rendering performance in frames per second (fps) when applying different see-through lens techniques based on data sets 1 and 2, evaluated from a close-up and a far perspective. Points are rendered as Point Splats.*

	Data Set 1		**Data Set 2**	
	Far	Close-Up	Far	Close-Up
#Rendered Points in Million	2.32	0.50	3.42	0.85
No technique applied	51.84	214.32	32.27	138.67
Simple Blending	49.65	210.84	30.39	135.84
Blueprints	48.18	208.31	29.84	134.76
Interactive Lenses	49.32	209.68	30.12	135.20
Halos	27.68	104.31	23.97	67.31

Table 3.3: *Rendering performance in frames per second (fps) when applying different see-through lens techniques based on data set 3, evaluated from a close-up and a far perspective. Points are rendered as Point Splats.*

	Data Set 3	
	Far	Close-Up
#Rendered Points in Million	4.85	1.04
No technique applied	23.01	108.63
Simple Blending	21.73	105.96
Blueprints	21.09	104.85
Interactive Lenses	21.46	105.21
Halos	16.92	52.57

Table 3.4: *Rendering performance in frames per second (fps) when applying different point-based rendering techniques based on data sets 1 and 2, evaluated from a close-up and a far perspective.*

	Data Set 1		**Data Set 2**	
	Far	Close-Up	Far	Close-Up
#Rendered Points in Million	2.32	0.50	3.42	0.85
GL_POINTS	86.39	378.07	60.02	246.12
Point Splats	51.84	214.32	32.27	138.67
Point Spheres	49.57	203.81	28.31	133.72
Silhouette Rendering	46.07	195.65	26.66	127.38
Solid Rendering	27.32	100.13	20.22	63.78
Combination 1 (Figure 3.5, row 3)	40.51	200.97	27.33	128.73
Combination 2 (Figure 3.5, row 4)	33.28	126.31	22.21	80.80

Table 3.5: *Rendering performance in frames per second (fps) when applying different point-based rendering techniques based on data set 3, evaluated from a close-up and a far perspective.*

	Data Set 3	
	Far	Close-Up
#Rendered Points in Million	4.85	1.04
GL_POINTS	40.24	194.35
Point Splats	23.01	108.63
Point Spheres	22.35	107.07
Silhouette Rendering	22.18	106.97
Solid Rendering	18.74	59.45
Combination 1 (Figure 3.5, row 3)	22.45	107.75
Combination 2 (Figure 3.5, row 4)	19.90	68.47

and 3.3). Since all techniques are implemented as screen-based post-processing effects, this stays true independent of the overall number of rendered points. Halos require a halo mask that is created by rendering corresponding surface categories twice, which effectively halves the average frame rate if halos are activated. However, interactive frame rates are still achieved for several million points being simultaneously rendered. Since the applied out-of-core rendering technique limits the number of rendered points per frame by dynamically selecting them, the proposed rendering approach can be applied to arbitrary large data sets while achieving real-time frame rates (Tables 3.4 and 3.5).

3.5 Conclusions

In this chapter, rendering approaches have been discussed that facilitate the exploration of enriched point clouds by taking into account per-point attributes as well as a point's relevance to a specific visualization or exploration task. In particular, geometric and

surface category information can be used to combine and configure different rendering techniques and color schemes to adapt the appearance of each point (i.e., its color, size, orientation, or shape) to reflect characteristics of the surface area represented by these points (e.g., solid, planar, non-planar, fuzzy). By applying image-based compositing based on G-Buffers, sophisticated focus+context visualization and interaction techniques, such as visibility masks or interactive lenses, are supported. As a remedy, the visual recognition of task-relevant objects within the point cloud depiction that are fully or partly occluded can be facilitated by masking out less relevant occluders. The proposed layered, multi-resolution kd-tree enables in addition to a spatial data selection a context-aware selection of LoDs. Hence, memory and processing resources can be used efficiently and adaptively.

The presented rendering approach offers many degrees of freedom for graphics and interaction design and adapts to a variety of use cases and exploration tasks from different domains. For future work, different research directions could be considered:

- Halo-based see-through lenses are currently considerably less efficient to render due to the costly preparation of the corresponding halo mask. By defining halos in screen space, that step could be avoided.
- The point density within point cloud depictions may vary heavily, not only due to density of the captured data itself, but also due to the nature of the out-of-core rendering and LoD selection. This can be addressed to an extent by scaling the point sizes based on the maximum loaded LoD in the corresponding subtree. A more precise calculation of appropriate per-point sizes –and thus a higher quality visualization– may be achieved by integrating rendering techniques that enable a per-frame reconstruction of object surfaces [127].
- The presented see-through lenses are just a selection of many focus+context techniques that have been proposed over the years to highlight occluded objects [58] and that might also be feasible in the context of 3D point clouds. In particular multiple views [77] or multi-perspective projection [118] techniques could be evaluated.

The presented rendering approach can be improved regarding rendering performance as well as interoperability with external GIS based on the techniques described in Chapters 4 and 5.

Chapter 4

Immersive Visualization of Point Clouds using VR Technology

This chapter addresses the immersive visualization of point clouds using VR technology, discussing the benefits of potential rendering optimization techniques as well as the applicability and usability of different interaction and locomotion techniques. The chapter is based in parts on the author's scientific publications in [3] and [8].

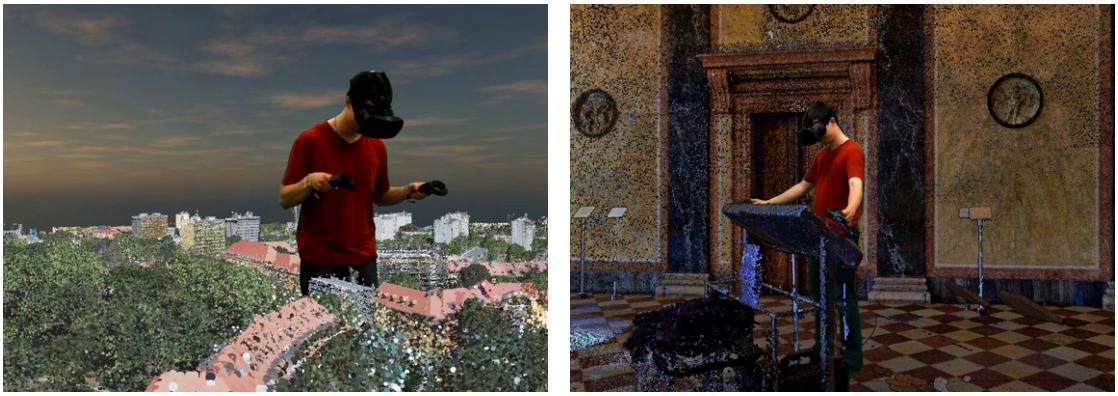
The following sections are structured as follows: Section 4.1 further motivates this chapter and gives a more detailed introduction. Section 4.2 and Section 4.3 discuss the advantages and disadvantages of different rendering optimization, interaction, and locomotion techniques. Section 4.4 describes setup and results of two-fold evaluation: The presented rendering optimization techniques are evaluated for real-world data sets with up to 2.6 billion points to show the rendering system's practicability and scalability. Furthermore, a pilot user study is conducted to gain initial insights into the usability of the presented interaction and locomotion techniques in the context of VR applications. Section 4.5 gives conclusions and outlines future research directions.

4.1 Introduction

VR devices, for example Oculus Rift[1] or HTC Vive[2], open up new ways to present digital 3D models on standard consumer hardware, granting users the perception of being physically present in a 3D virtual environment [12, 24]. In general, the corresponding 3D models can be designed and modeled for a particular purpose (e.g., game environment) or can be derived by captured data from real-world sites or assets (e.g., building models). For complex sites, e.g., buildings with a highly detailed interior, or large areas, e.g., cities and landscapes, manually modeling 3D contents is neither time efficient nor cost efficient due to the required effort [106]. As a remedy, there is a strong demand for methods and techniques that (1) automatically and efficiently capture real-world sites of arbitrary size and complexity with high precision and that (2) directly integrate the resulting 3D contents into VR applications without having to sacrifice any captured details.

In recent years, automatically capturing real-world sites by means of point clouds has become increasingly cost efficient and time efficient due to technological advances

[1] https://www.oculus.com/rift/
[2] https://www.vive.com



(a) *Airborne scan of a city.* (b) *Terrestrial indoor scan.*

Figure 4.1: *Examples of point clouds being immersively visualized using the rendering system presented in this chapter and an HTC Vive. Supported interaction techniques include measuring of distances as well as rotating and scaling of the rendered data.*

in remote and in-situ sensing technology (Chapter 2). As presented in Chapter 3, point clouds of arbitrary size and density can be directly used as interactively explorable models by combining LoD concepts, out-of-core strategies, and external memory algorithms. In combination with web-based rendering concepts, these external memory algorithms allow the interactive inspection and visualization of point clouds on a multitude of devices featuring vastly different CPU and GPU capabilities (Chapter 5). However, common rendering systems for point clouds typically focus on non-immersive applications, carefully balancing the trade-off between rendering quality and performance [162]. In VR applications additional challenges are raised:

- **Stereo rendering**. To generate a stereoscopic image, each scene must be rendered for two displays simultaneously.
- **High rendering quality**. Visual artifacts such as visible holes between neighboring points or visual clutter tend to be more noticeable on VR displays, can easily break the immersion [160] and, therefore, need to be fixed.
- **High frame rates of 90 fps**. Nausea, i.e., the feeling of motion sickness, typically occurs when the motion-to-photon-latency, i.e., the time required for the depicted images to update after a physical movement by the user, becomes too high. As a remedy, the built-in displays of VR devices such as Oculus Rift or HTC Vive operate at 90 Hz [177]. Hence, frames must be rendered at a considerably higher speed compared to non-immersive applications, for which frame rates between 30 and 60 fps are usually sufficient.

For these reasons, rendering systems must be optimized with respect to two conflicting goals: An improved visual quality of render artifacts and an increased rendering performance. Furthermore, VR applications require sophisticated locomotion techniques to efficiently navigate through a 3D virtual environment without causing nausea or loss

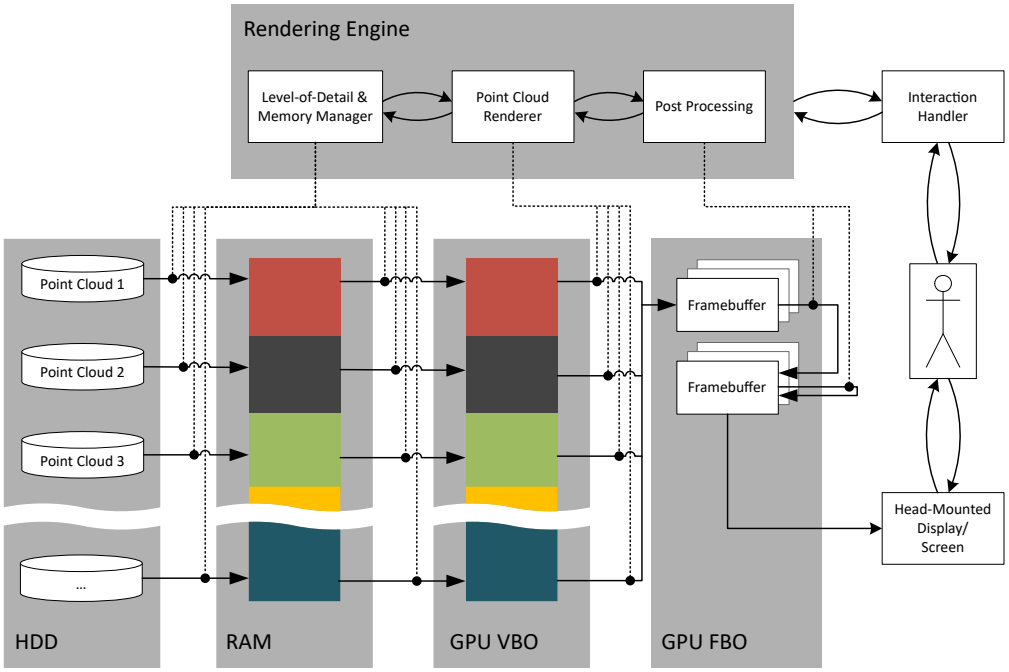


Figure 4.2: *Rendering pipeline and data flow between hard disk drive (HDD), random-access memory (RAM), VBO, and FBO. A common memory budget is shared by all rendered data sets.*

of orientation. Intuitive interaction techniques, such as measuring and annotating within point cloud depictions, that have minimal impact on the rendering performance are also required. In the following sections, a rendering system is presented that allows for an immersive, nausea-free exploration of point clouds on state-of-the-art VR devices (Figure 4.1). To that end, *performance optimization techniques* that speed up the rendering pipeline are combined with *image optimization techniques* that improve the overall image quality. To enable an in-depth exploration and inspection, different *interaction techniques* are provided, such as, to measure distances and areas or to scale and rotate visualized data sets, as well as several *natural* and *artificial locomotion techniques*, that can be selected and configured at runtime.

4.2 Rendering Optimization Techniques

The rendering system presented in this chapter shares some similarities to the context-aware point-based rendering system presented in Chapter 3: A multi-pass pipeline (Figure 4.2) is used to seamlessly combine three distinct render stages: (1) Selecting subsets of representative points, (2) rendering those subsets, and (3) applying image-based post-processing on the resulting render artifacts.

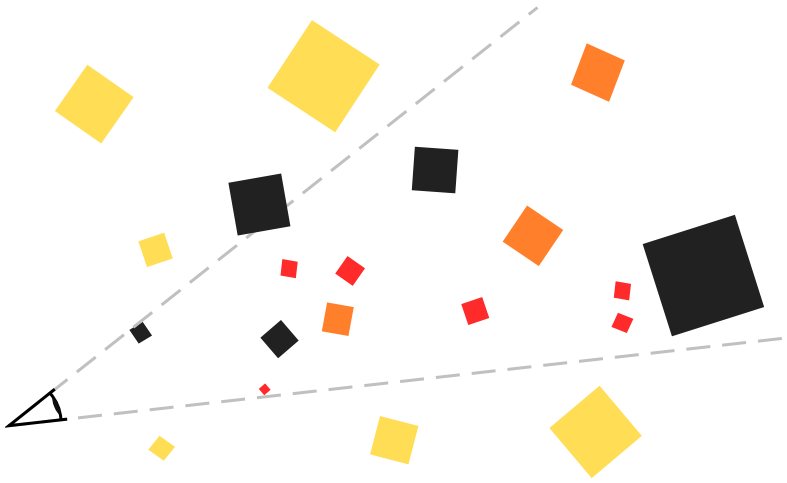


Figure 4.3: *Culling techniques used to reduce the amount of points to be rendered: View frustum culling (yellow), occlusion culling (orange), detail culling (red).*

Level-of-Detail and Data Subset Selection

Rather than rendering every point of a given data set, a representative subset of points is determined on a per-frame basis that can be managed by available CPU and GPU capabilities. To that end, two major criteria are taken into account (Figure 4.3): First, points outside the current view frustum are excluded as they would not be visible anyway (i.e., *view frustum culling*). Second, points are aggregated based on their spatial position to accommodate for the perspective distortion resulting in areas farther away from the current view position to appear smaller on screen (i.e., *detail culling*). To provide an efficient access to representative data subsets, the point cloud is hierarchically subdivided using a kd-trees. As discussed in Chapter 2, this allows for minimal tree traversal times during rendering. For each point cloud a separate kd-tree is generated in a pre-processing step. A flexible memory budget is defined to limit the number of points that can be rendered per frame. While each point cloud is rendered separately, the memory budget is shared among them. As the performance may vary based on scene complexity and applied rendering techniques, the memory budget is adjusted dynamically to guarantee 90 fps at any time.

Data Subset Rendering

Selected representative subsets are rendered into G-buffers combining 2D textures for, color, depth, and normal values. This provides efficient means to combine image-based post-processing techniques in the subsequent render stage that improve the visual quality of the final image being displayed on the VR device. Furthermore, different point-based rendering techniques can be dynamically configured, selected, and combined at that render stage. This makes it possible to (1) dynamically adapt the appearance of each point to the current exploration task (Chapter 3) control the overall rendering performance

(Section 4.2.1). Contrary to the rendering system presented in Chapter 3, the G-Buffers do not contain any ID textures since interaction techniques evaluated in the context of this chapter focus on measuring and transformation tasks, that require no data layers other than a point's spatial position 4.3.

Image-Based Post-Processing

The concluding image compositing stage of the rendering pipeline operates recursively on the previously generated G-Buffers, allowing to configure and combine several image-based rendering techniques. As an example, rendering techniques for hole-filling, blurring, anti-aliasing as well as edge detecting and highlighting can be efficiently combined to improve the visual quality of the final rendered image (Section 4.2.2).

4.2.1 Performance Optimization

To further improve the performance of the presented rendering system on state-of-the-art VR devices, the following rendering techniques are implemented and evaluated:

Hidden Mesh Rendering

Due to the radially symmetric distortion produced by the lenses of an VR device, the actually visible area of the built-in screens is restricted to a circular area (Figure 4.4). To prevent unnecessary fragment shader operations, fragments outside that area are discarded early, using a separately rendered mesh representing the hidden parts of the screen as a mask that is evaluated using early fragment testing [177].

Reverse Painter's Algorithm

As a GPU-based occlusion culling technique (Figure 4.3), the reverse painter's algorithm [78] describes efficient means to prevent occluded fragments from being unnecessarily processed by the fragment shader. Based on early fragment testing, scene objects should be rendered in order of their distance to the view position for the technique to have a measurable effect. Calculating such an order on a per-point basis would be inefficient. As each point belongs to a specific node of the kd-tree however, that calculation can instead be performed on a per-node basis, considering only those nodes that have been selected for rendering.

Single-Pass Stereo Rendering

VR devices require to render all view dependent items from two different views representing the left and right eye, respectively. *Single-pass stereo rendering* aims to reduce the *CPU overhead* by rendering both views in a single render pass [82]. To that end, the frame buffer size is doubled, assigning each half to one eye. Instanced rendering is used to avoid duplicated draw calls. It duplicates each point and applies the corresponding view transform at the vertex shader stage. To minimize the probability of points spilling over

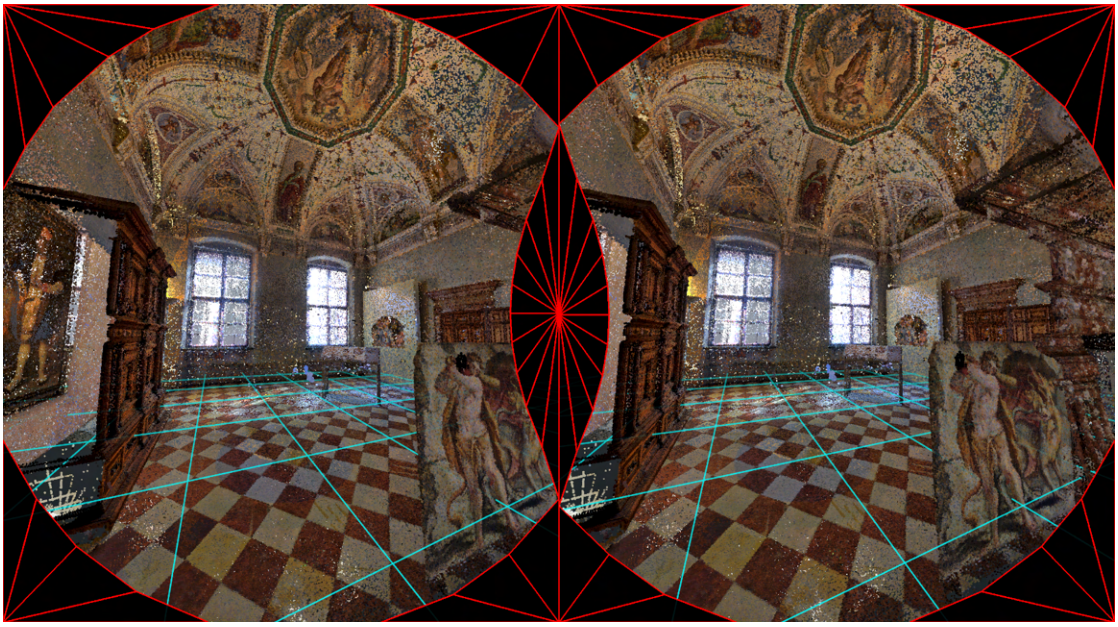


Figure 4.4: *A separately rendered mesh serves as a mask to discard fragments beyond the visible area of an VR device's screens early on.*

into the opposite half of the frame buffer, a heuristic is applied that shrinks points close to the border. Preventing such artifacts completely would require to discard affected fragments explicitly, which would be incompatible to early fragment testing as required by the techniques presented above.

4.2.2 Image Optimization

The immersiveness of a virtual scene is negatively affected by any kind of visual artifacts or inconsistencies one would not expect in the real-world, such as aliasing, z-fighting, and insufficient or missing depth cues [12]. In point cloud depictions, the most noticeable artifacts arise when points representing a continuous surface are sized inappropriately, resulting in either a holey appearance of those surfaces or visual clutter due to overlapping points (Figure 4.5 (a+b)). To minimize such artifacts the following rendering techniques are implemented and evaluated:

Adaptive Point Sizes

The different nodes of LoD data structures exhibit noticeable differences regarding the point density. Thus, assigning all points a uniform size results in either holes between neighboring points or overlaps and visual clutter (Chapter 3). Schütz [160] addresses that issue by adjusting each point's size based on the maximum LoD within its local neighborhood. Point sizes are also adjusted adaptively by the rendering system presented in this chapter. However, the corresponding rendering technique operates on a per-node

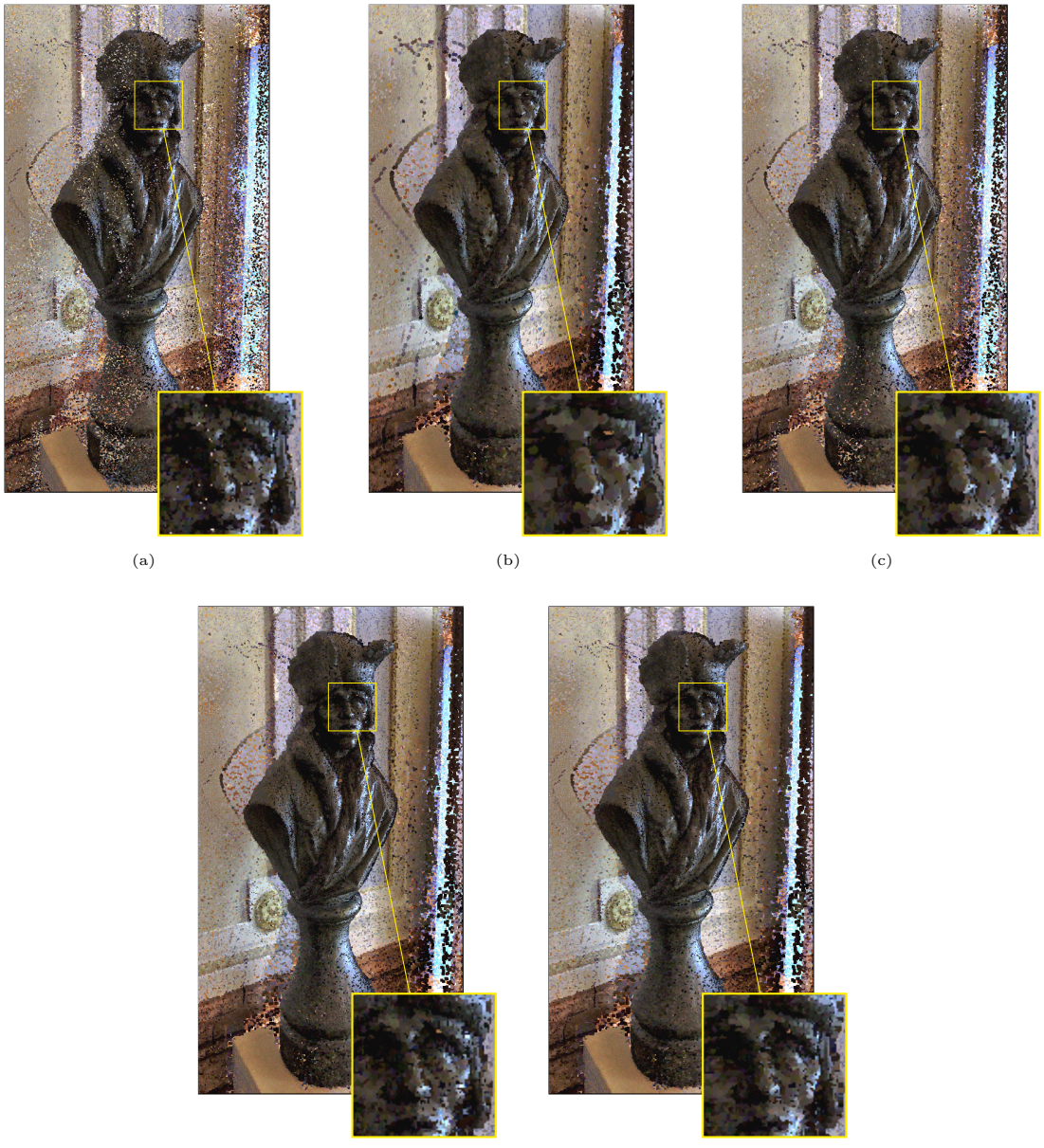


Figure 4.5: *Incorrectly sized points may lead to a holey appearance (a — point size of 1px) or visual clutter (b — point size of 5px). An adaptive point size strikes a balance between both artifact types, but does not eliminate them completely (c). This can be minimized by applying paraboloid rendering (d — diameter of 5px) or filling (e — 5x5 filter kernel and point size of 1px).*

Figure 4.6: *Contrasting color values can be harmonized using blurring to smooth aliasing and z-fighting.*

instead of a per-point basis, thus, avoiding the need for a separate render pass to calculate each point's LoD. In that regard, the implemented rendering technique is similar to the one proposed by Scheiblauer and Wimmer [157]. However, it uses inherently balanced kd-trees in favor of octrees.

For each node, its deepest descendant that has been selected for rendering is determined. The adaptive point size for that descendant is then applied to all of its ancestors. Furthermore, point sizes are calculated based on a node's bounding box rather than its LoD since nodes of the same LoD might still feature drastically different point densities. While this approach drastically and effectively reduces holes and overlaps, it does not exclude those artifacts entirely. For example, if nodes selected for rendering form a heavily unbalanced tree, some points might be rendered too small (Figure 4.5 (c)). The resulting holes are filled via post-processing.

Paraboloid Rendering

Paraboloid rendering is a technique introduced by Schütz and Wimmer [162] that aims to further reduce visual clutter by rendering points not as flat, screen-aligned disks but as paraboloids oriented towards the view position. By adding a depth offset to fragments based on their distance to the corresponding point's center, undesired occlusions are drastically reduced (Figure 4.5 (d)). However, as this technique requires to modify depth values at the fragment shader stage, it is incompatible with early fragment testing and, thus, with most of the performance optimization techniques discussed in Section 4.2.1.

Post-Processing

Several post-processing techniques are combined to further improve the visual quality: *Screen space ambient occlusion* (SSAO) [104] and *eye-dome lighting* (EDL) [29] add depth cues and highlight silhouettes, blurring [96] smoothes aliasing and z-fighting (Figure 4.6).

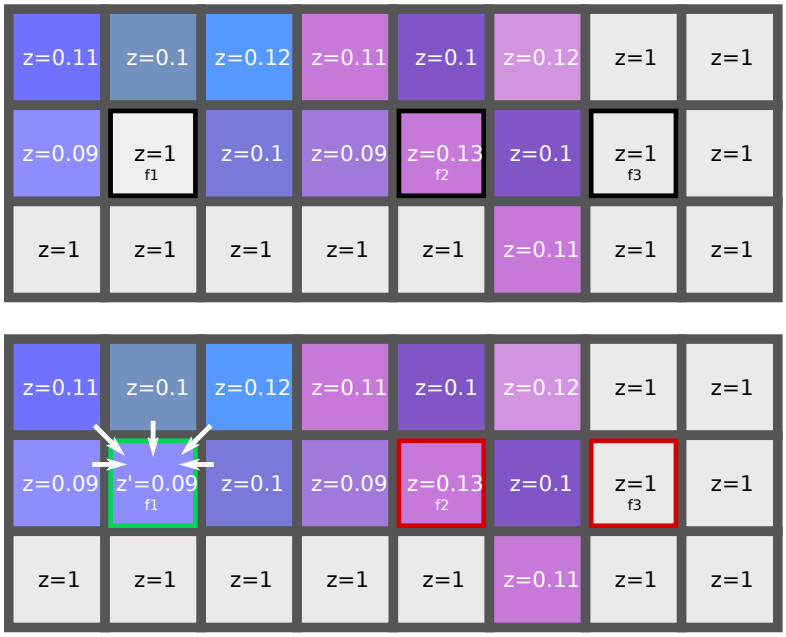


Figure 4.7: *Fragment* f1 *is detected as a hole based on depth differences to its neighbors and gets assigned the minimum depth value within its neighborhood;* f2 *and* f3 *remain unchanged as they fail the distance threshold and the minimum number of neighbors, respectively.*

Furthermore, remaining holes between points representing the same surface are filled (Figure 4.5 (e)). To that end, the technique presented by Dobrev *et al.* [52] is adapted, applying two one-dimensional filter kernels instead of a single two-dimensional one for a performance speed up. The filter kernel checks a pixel's neighborhood for significant depth differences and overwrites corresponding pixels with interpolated values from those neighbors being closest to the view position (Figure 4.7).

Multisampling

An image optimization technique to smooth aliasing and reduce z-fighting even further would be multisampling, which provides a smoother color transition between neighboring fragments by sampling them several times. While this technique also reduces the visibility of outliers, it also requires rendering fragments several times, thus, drastically affecting the performance, especially when combined with post-processing effects. As a remedy, this image optimization technique was ultimately rejected in the context of this chapter.

4.3 Interaction and Locomotion Techniques

Handling of user interaction as well as updating the point cloud depiction accordingly is managed by a separate component of the presented rendering system, the *interaction handler* (Figure 4.2). This component's tasks include (1) configuring and selecting applied

rendering techniques, (2) locomotion, (3) measuring distances and areas, as well as (4) transforming rendered point clouds.

When required, the interaction handler is also able to communicate with the rendering and post-processing stage. An example of this is the interactive selection of points for measurement or transformation tasks. To minimize the performance impact of corresponding interaction technique, a separate rendering pass is introduced, that centers the view at the corresponding controller's location while looking alongside the pointing direction. Using an orthographic projection, the scene is then rendered into a separate G-Buffer storing a single depth texture. Combined with the known location of the controller, the stored depth values provide efficient means to calculate the targeted point's position. The idea of a perspective projection was rejected to ensure a constant precision of the calculated positions independent of the distance to the view center.

4.3.1 Interaction

In the context of this chapter three different interaction techniques that enable users to explore and manipulate rendered point clouds are implemented and evaluated. These techniques are an imprecise yet intuitive *virtual tape measure*, a *precise measurement tool* that allows querying distances and areas between dynamically selected points, and a *gesture-based transformation tool* providing means to scale and rotate data sets. All interaction techniques can be controlled with HTC Vive and Oculus Rift interchangeably, requiring only the corresponding motion controllers.

Virtual Tape Measure

The virtual tape measure is activated by simultaneously pressing the grip buttons on both controllers. When active, a yellow rectangle with equidistant markings is rendered between the controller models, while the measured length is constantly displayed above that rectangle (Figure 4.8). Text and tape measure are rendered using procedural textures and distance fields [70] to prevent visual artifacts when viewed up close. The length displayed is calculated in object space. This enables users to measure objects that usually would not fit in an arm span, such as the height of a church tower or the length of a house, by rescaling the data beforehand. Both, the grabbing motion and the rectangle's appearance, contribute to the virtual tape measure resembling its real-world counterpart. In the sense of an interface metaphor [99], this resemblance enables users to transfer their knowledge of using real-world tape measures into the virtual world, making this technique easier to use. While being a quick and intuitive way to measure distances between two positions, the virtual tape's precision heavily depends on the user's ability to keep his hands steady. Unlike the precise measurement tool, it also fails provide functionality to measure area sizes.

Figure 4.8: *The virtual tape measure enables the user to take quick measurements in object space.*

Precise Measurement Tool

The *precise measurement tool* (Figure 4.9) allows dynamically selecting points to measure exact distances as well as the areas. The points are selected by pointing at them and pressing the trigger button, using the orthographic, projection-based approach described at the beginning of this section. This interaction technique provides two different modes:

1. In *distance mode*, selected points are connected by a line, and a label indicating its length is displayed next to it. After every two points the measurement is automatically completed. Selecting an additional point will start a new measurement.

2. In *surface mode*, selected points form a polygon and its area and the lengths of its outer edges are displayed. The first three selected points define a plane into which every subsequent point is projected. All projected points form a surface that is reconstructed upon each new selection, using either a *Bowyer-Watson triangulation* [93] or an *advancing front reconstruction* [101]. The latter handles concave polygons better, but often leads to unintuitive results, especially when there is a large variance in edge length. All points are projected into a common plane to compensate for unavoidable inconsistencies during the data acquisition process, since even perfectly even surfaces will usually not be completely planar [57].

Similar to the virtual tape measure, procedural textures and distance fields are applied to render distance and area labels. Pressing the left grip button starts a new measurement while the previous lines and polygons are still displayed. The right grip button cancels the current measurement, deleting previous measurements from last to first

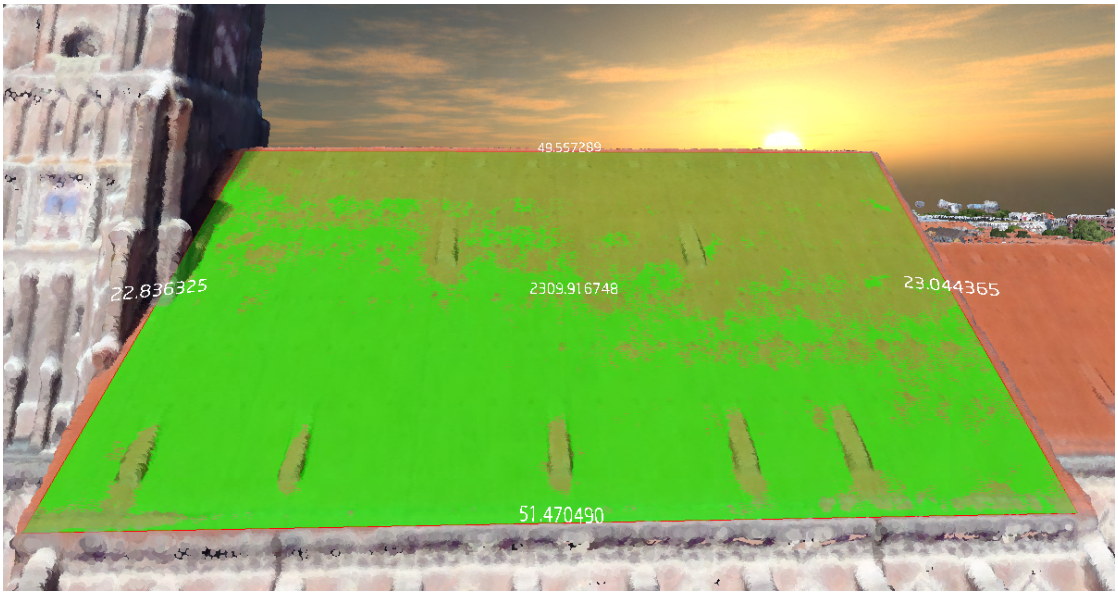


Figure 4.9: *The precise measurement tool allows measuring both distances and areas by point selection.*

with every subsequent button press. A dedicated menu pane allows switching between distance and surface mode and choosing the method for the surface reconstruction.

Gesture-Based Transformation Tool

The gesture-based transformation tool (Figure 4.10) is operated in a analogous way as equivalent tools from other VR applications as Tilt Brush by Google[3] or Oculus Quill[4]. Objects can be grabbed with one or both controllers by pressing the respective grip buttons. The grabbed objects can then be manipulated by performing gestures with the grabbing controllers. In this specific case, the following gestures are supported:

1. **Translate**: Real-world movement of the controller is directly translated to movement of the point cloud. To the user, the point cloud appears fixed to the moving controllers.
2. **Scale**: Moving both controllers apart or bringing them closer together scales the point cloud up or down.
3. **Rotate**: Rotating both controllers around an imaginary point in between the controllers also rotates the point cloud. However, to prevent nausea, this rotation is limited to the up axis.

While scaling and rotating requires the use of both controllers, translating can also be triggered with just one controller.

[3] `https://www.tiltbrush.com/`
[4] `https://www.oculus.com/experiences/rift/1118609381580656/`

Figure 4.10: *The gesture-based transformation tool allows to scale, translate and rotate point clouds interactively.*

4.3.2 Locomotion

Users can navigate using different locomotion techniques: Locomotion based on *gamepads and keyboards*, *real walking*, *joystick flying*, *point & teleport* (P&T), or *dashing*:

Keyboard/Gamepad Locomotion

Depicted point clouds can be navigated using a gamepad or a keyboard. On gamepads the left thumbstick controls the movement relative to the gaze direction. The speed of the movement can be controlled by the inclination of the stick, with no movement in the center position and maximum velocity when the stick is completely pushed in one direction. The right thumbstick enables rotating (left and right) or changing altitude (up and down) in the virtual world. The gamepad's right trigger button accelerates any movement. Keyboard controls are similar, with the WASD keys controlling movement and the arrow keys controlling rotation and tilt.

Real Walking

Position and orientation reported by the tracking system are taken into account so that physical movements are translated to equivalent motions in the virtual world. This enables running and walking, but also movements like bowing down (e.g., to inspect details of an object) crouching, sitting, or lying prone on the ground.

Joystick Flying

The motion controller projects a green ray that indicates the movement direction. Pressing the trigger button allows movement along that direction in the virtual world. The movement speed increases linearly with the force of the trigger pull. Any changes to

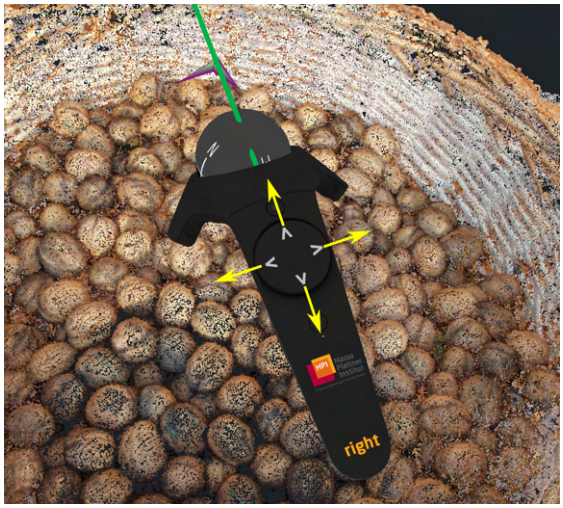


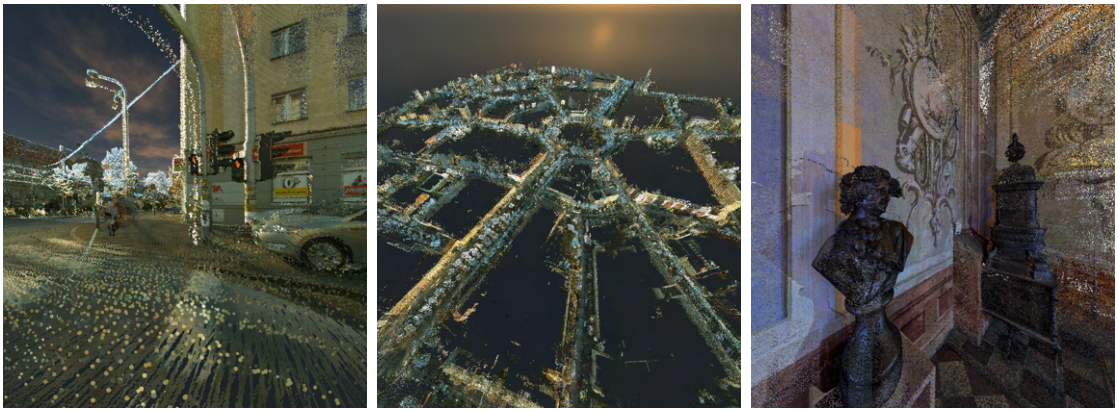
Figure 4.11: *The touchpad allows movement in the selected direction relative to the forward direction symbolized by the green ray.*

direction and speed of the movement are applied directly, conforming to the Oculus best practices guide[5] that highly discourages gradual accelerations. While the trigger is being pressed, the direction of movement is strictly forward, symbolized by the green ray. The decision in favor of that behavior was made due to an observation by Bozgeyikli *et al.* [32] that humans tend to primarily move forward and rarely backward or to the side. Still, users may explicitly move backwards or sideways in relation to the current forward direction by using the controller's touchpad (i.e., for HTC Vive) or analogue stick (i.e., for Oculus Rift) (Figure 4.11). Again, the movement speed increases linearly with growing distance to the touchpad's center and the analogue stick's inclination, respectively.

Point & Teleport Locomotion

Based on a study by Bozgeyikli *et al.* [32], recommending that technique explicitly for applications with an explorative nature, P&T enables an instantaneous movement to interactively selected positions in the virtual world. In addition to the green ray used in joystick flying mode, a red cross is rendered onto the ray, indicating the position of the user's feet after completing the teleport. The cross on top of the ray can be repositioned (i.e., shifted further away or pulled closer in) using a controller's touchpad or analogue stick. This target selection diverges from other P&T implementations and current teleportation-based VR games (e.g., *RoboRecall*[6]) in which the targeted position always snaps to the ground of the geometry. Due to the very nature of a point cloud the generation of a corresponding ground mesh would require an additional preprocessing step. We avoid such a preprocessing step by allowing users to modify both, position and altitude above the perceived ground, as long as those modifications do not result in a

[5] https://static.oculus.com/documentation/pdfs/intro-vr/latest/bp.pdf
[6] https://www.epicgames.com/roborecall/en-US/home



(a) *Pedestrian view of a mobile mapping scan.*

(b) *Birds-eye view of a mobile mapping scan.*

(c) *Close-up view of a terrestrial indoor scan.*

Figure 4.12: *Scenes used during the performance evaluation.*

position outside of the point cloud's bounding box. The teleport is initiated by a single press of the trigger button. The forward direction is not changed, as the ability to do so was determined as a possible cause for disorientation by Bozgeyikli *et al.* [32].

Dash Locomotion

The dash locomotion is a variation of P&T featuring the same user interface elements and limitations (i.e., regarding the bounding box). Pressing the trigger button initiates the transport process, locking locomotion controls and moving the user in a fast and sudden gliding motion to the new position. On arrival at the new position, the controls are unlocked once more for further movement. As recommended by the Oculus best practices guide, the acceleration is instantaneous and not gradually. It is claimed that short bursts of movement with a high, constant velocity are less likely to cause nausea, as the vestibular system can only detect acceleration and not speed.

4.4 Performance Evaluation and Usability

The presented rendering system has been implemented on the technological basis of C++, OpenGL, GLSL, and OpenVR[7]. The test system featured an Intel Core i7-5820K CPU, 16 GB main memory (DDR4, 1200 MHz), a GeForce GTX 980 with 4096 MB device memory(GDDR5, driver version 390.77) as well as an HTC Vive as the output device. Measurements on an Oculus Rift lead to comparable, slightly better performance due to the tighter view frustum.

4.4.1 Rendering Performance

The test data sets comprised a mobile mapping scan of an urban area (2.6 billion points) and a terrestrial indoor scan of an individual site (1.5 billion points). The performance

[7] `https://github.com/ValveSoftware/openvr`

Table 4.1: *Average rendering performance of performance optimization techniques in ms/frame. All test runs include view frustum and detail culling. Dynamic memory budget was disabled to ensure comparability of measured values.*

	Scene 1	Scene 2	Scene 3
#Rendered points	19.8M	6.9M	11.6M
Default	15.93ms	9.23ms	12.15ms
Hidden Mesh	15.59ms	9.19ms	11.87ms
Reverse Painter's	12.95ms	9.27ms	11.11ms
Single-Pass Stereo	17.48ms	9.82ms	13.54ms

evaluation was conducted for three different scenes (Figure 4.12): A close-up and a zoomed out view of the urban area (Scene 1 and 2) as well as a close-up view of the individual site (Scene 3). The dynamic memory budget, which guarantees the constant frame rate of at least 90 fps, was disabled for the evaluation to ensure the comparability of the measured values.

Both, the *hidden mesh* and the *reverse painter's* algorithm, improve the rendering performance. However, their effectiveness varies, depending on the number of affected fragments (Table 4.1). *Single-pass stereo rendering* proved to be less effective as the primary rendering bottleneck is the GPU, not the CPU. On the contrary, the technique even slows the rendering pipeline as view frustum culling needs to be combined for both eyes, thus notably increasing the amount of unnecessarily rendered points per side. Regarding image optimization techniques, *paraboloid rendering* and *multisampling* —as expected— significantly reduces the rendering performance (Table 4.2) and thus should only be used if the z-fighting becomes too prominent and significantly affects the immersion. On the other hand, post-processing effects and adaptive point sizes only have a moderate performance impact. While combining all post-processing techniques would amount to a significant performance drop, doing so will hardly be necessary. As an example, *EDL* and *SSAO* aim for similar effects, whereas *blurring* will only be noticeable in specific scenes, e.g., if color values of neighboring points are inconsistent due to an erroneous capturing process.

4.4.2 User Study Setup

To evaluate the usability of the presented interaction and locomotion techniques, a pilot user study featuring a small number of participants was conducted. The tendencies discovered in that pilot user study should be seen as the basis for future hypotheses investigated by full-fledged user studies with more participants in future work.

Table 4.2: *Average rendering performance of image optimization techniques in ms/frame. For paraboloids, hidden mesh rendering and the reverse painter's algorithm were deactivated and an oversized point size (5 px) was used. Dynamic memory budget was disabled to ensure comparability of measured values.*

	Scene 1	Scene 2	Scene 3
#Rendered points	19.8M	6.9M	11.6M
Default	12.82ms	9.21ms	10.77ms
Adaptive Pt. Size	13.88ms	9.48ms	12.46ms
SSAO		+ 2.67 ms	
EDL		+ 0.32 ms	
Filling		+ 1.07 ms	
Blurring		+ 2.17 ms	
Multisampling	17.91ms	10.14ms	16.94ms
Paraboloids Def.	12.72ms	10.77ms	10.26ms
Paraboloids	15.17ms	18.45ms	15.62ms

Participants

The participants were eight students aged 18 to 21, six male and two female. All had a background in computer science, but only four had used VR devices before. Of these participants, only one had sporadic contact with VR technology while the remaining three made such contact solely via other user studies.

Experimental Session and Measures

Participants began with a training session introducing and explaining each technique by setting simple tasks like following a straight line, scaling up and down and measuring a rooftop while within the virtual environment.

The first actual experiment was set within a terrestrial indoor scan. Participants were asked to follow a predefined path while using the gamepad locomotion. The path was 80 meters long including various sharp turns and narrow doorways. After completion the participants were asked to fill a questionnaire about the locomotion technique used. They could rate the technique on a 5-point scale regarding intuitivity, ease of use, and comfort. They were also asked about signs of discomfort and positive or negative remarks. Trial and questionnaire were repeated for the remaining three locomotion techniques (joystick flying, P&T locomotion, dash locomotion). Afterwards, the participants had to complete a last pass, but this time were allowed to freely choose a technique and were asked to state the reasons for that choice on the questionnaire. The second experiment was set within a city-wide aerial scan with wider spaces and a longer path of 945 meters with long straights (Figure 4.12 (a+b)), following the same procedure.

For the third experiment, all locomotion techniques beside real walking were disabled.

Participants were placed in front of a church within the city-wide aerial scan and tasked to measure the length of its roof using the virtual tape measure. To complete this task, they had to use the gesture-based transformation tool to scale the scenery down beforehand. They were asked to fill a questionnaire afterwards, rating the tape measure on a 5-point scale regarding intuitivity, ease of use and precision. This procedure was repeated with the precise measurement tool. Afterwards, participants were asked to measure the size of the church's tower with a technique of their liking and to give reasons for their choice. A last questionnaire was given to grade the gesture-based transformation tool on a 5-point scale regarding intuitivity, ease of use, and usefulness as well as to comment on the technique's overall comfort.

4.4.3 User Study Results

The pilot user study provided a number of preliminary insights about the implemented interaction and locomotion techniques. A brief discussion of what conclusions can be drawn from these insights is given at the end of this section.

Gamepad Locomotion

Gamepad locomotion had the second highest score in almost all categories in the first and the second experiment. Only for the indoor scene it came third in the *comfort* category, surpassed by joystick flying and dash locomotion. Two participants reported nausea and dizziness. Participants made positive remarks about the familiarity with the controller mapping known from video games and the ability to turn in the virtual environment without having to turn around physically. They further praised the possibility for a fluid and fast motion enabling them to turn without having to stop. On the downside, they reported problems with the height control, due to it being mapped on the same stick as turning and, thus, being easy to trigger accidentally. One participant stated a feeling of disorientation when the body was moved simultaneously with gamepad input.

Joystick Flying

Joystick flying had the highest score in all categories in the first and the second experiment. One participant reported nausea, another experienced a short loss of balance in a fast turn that forced him to do a small side step to keep his balance. The participants made positive remarks about being able to change the speed with the trigger pull though some criticized it as too sensible. The control over the direction by pointing, allowing immediate corrections and course changes, was mentioned as positive as well. The necessity to turn the body to face the current movement direction faced a mixed reception: Some participants experienced it as natural while others found it cumbersome. No participant made extensive use of the touchpad for movement.

P&T Locomotion

P&T locomotion was ranked third regarding *intuitivity* and fourth regarding *comfort* in both, the first and second experiment. Its *ease of use* was ranked third (outdoor) and lowest (indoor), respectively. Participants commented that the technique was a fast way to travel long distances. One participant reported nausea, whereas another one stated that it felt less unnatural and that he therefore felt less dizzy while using it. However, the necessity to place the cross on floor height to keep the current altitude and the repeated need for corrections was perceived as very negative. While some participants experienced the technique as merely cumbersome, others reported that they repeatedly teleported to other locations than expected. The mandatory pause to adjust the position of the cross was commonly perceived as disrupting.

Dash Locomotion

The *intuitivity* of the dash locomotion was ranked fourth (indoor) and third (outdoor), respectively. Its *ease of use* was ranked third (indoor) and fourth (outdoor), its *comfort* second (indoor) and third (outdoor). One participant reported nausea as well as a light headache in the second experiment. The participants perceived the dashing motion as positive, especially in comparison with P&T. An important aspect for the participants was that they could see where they were going and did not have to regain their orientation afterwards. It also encouraged them to move by many smaller increments instead of trying to cover large distances by pushing the target far out. However, the main critique points of the P&T locomotion (i.e., the red cross and its necessary adjustment) were also present with the dash locomotion.

Virtual Tape Measure

Compared with the precise measurement tool, the virtual tape measure ranked first regarding *intuitivity* and *ease of use*, but received a lower ranking regarding its *precision*. The participants had no problems using the technique, even though a few participants remarked that they would like to see both ends of the rendered tape for a more precise measurement.

Precise Measurement Tool

The scores of the precise measurement tool were lower than that of the virtual tape measure regarding *intuitivity* and *ease of use*, but significantly higher regarding *precision*. Overall, the participants had no major issues with the technique, but noted that the required pressure on the trigger sometimes resulted in them missing the point they intended to select.

Gesture-Based Transformation Tool

On the 5-point scales for *intuitivity*, *ease of use* and *usefulness*, the technique scored averaged values between 3 and 4 points. Participants found it easy to change the scale of the scenery but commonly struggled with the grip button's location and with moving the scaled model around. Some participants experienced an accidental translation or rotation of the model due to not keeping the controllers in a straight line, crossing them, or moving them up or down while scaling.

Preferences: Locomotion

When left with a choice, most participants chose the joystick flying, both in the indoor (five participants) and the outdoor scene (six participants). Reasons given were that they would not have to stop for adjustments, could see where they were going, and had at every moment control over speed and course. Fun was also a factor, as was not only noted quite often on the questionnaire, but could also be observed in the study: Participants commented that the application was missing a "superhero soundtrack", compared it with a fast ride on a motorbike or seemed outright disappointed when the end of the path was reached and another compared it with a fast ride on a motorbike. However, some participants also chose the gamepad locomotion technique (two times for the indoor scene, two times for the outdoor scene). Reasons given were the ability to turn without body involvement via the analogue stick, as well as the control scheme being similar and thus, familiar to video games. While P&T locomotion was not chosen, one participant opted to use dash locomotion in the indoor environment for its precision and fluid movement.

Preferences: Measurement

Six participants preferred the precise measurement tool while only two opted for the virtual tape measure. Reasons given were superior precision, the possibility to move between selections, and the fact that most measurements could be done without the necessity to scale the scenery. The two participants who selected the virtual tape measure stated that it gave them more flexibility in the measurement, as they found it difficult to select precisely the point they intended to select.

Discussion

The low ranking scores of P&T locomotion is noteworthy, as P&T and joystick flying were also compared in the study by Bozgeyikli *et al.* [32]. In their study, the difference in score between the two locomotion techniques was much less significant. The most likely explanation for that discrepancy is the modified target selection process in the context of this chapter, i.e., selecting the target position via the red cross rather than the location snapping to the perceived ground. For the future study, the current implementation of P&T locomotion could be modified with an additional rendering pass similar to the precise measurement system, albeit at a slightly decreased rendering performance. However, this would also restrict movement as users could only navigate from point to point: Vertical

movement would be very limited and point clouds featuring a particular low density could be hard to navigate. The dash locomotion technique should be altered similarly, as the scores of this pilot study indicate it being a more comfortable, less disorienting version of P&T, as long as the issues with the target selection can be solved.

Another noteworthy tendency is the popularity of the gamepad locomotion as studies, as the one done by Sarupuri *et al.* [155], indicate that users prefer locomotion techniques based on 6DOF input devices. This might be a side effect of the computer science background and the relatively young age of the participants as they are likely to have grown up with video games and are very familiar with gamepads as input devices. This should be examined closely in the later study and therefore participants from various age groups should be selected. Joystick flying proved to be the preferred locomotion technique. The virtual tape measure, the precise measurement tool and the gesture-based transformation tool performed as expected.

4.5 Conclusions and Future Work

The presented rendering system enables users to interactively explore point clouds on consumer-level VR devices, providing immersion-preserving visual quality and frame rates that avoid nausea at all times. To this end, various rendering, interaction, and locomotion techniques are provided that can be freely combined and configured at runtime using a multi-pass rendering approach, offering many degrees of freedom for graphics and application design. As a remedy, the presented rendering approach promises to be highly beneficial for applications in the fields of digital documentation, preservation, and presentation of natural and cultural heritage as it allows users to remotely explore and inspect digital twins of endangered or hardly accessible sites in a much more immersive way than existing solutions [98]. In building information modeling or urban planning and development, it facilitates planning processes by providing efficient means to integrate additional, mesh-based geometry such as 3D floor plans or building models into the generated stereoscopic point cloud depictions. Performance tests on several massive data sets with up to 2.6 billion points show the feasibility and scalability of the rendering system. Results of a pilot study indicate the usability of the provided interaction and locomotion techniques.

As future work, full-fledged user studies featuring more participants with diverse backgrounds should be conducted to evaluate and solidify the tendencies discovered in that pilot user study. To further improve the rendering performance, distributing the stereo rendering across two separate GPUs should be considered and evaluated as proposed by Vlachos [178]. To support hardware that is not specifically designed for VR, web-based rendering concepts for thin clients, that drastically reduce client-side hardware requirements by using a central server to render and distribute stereoscopic panoramas, may be combined with the presented techniques as described in Chapter 5. Additional future work could focus on combining eye tracking technology with VR

technology. Manufacturers of eye tracking solutions such as Tobii[8] and Pupil labs[9] are already producing integration kits for the HTC Vive, while the Oculus Half Dome prototype[10] comes with eye tracking technology already installed. Such solutions would enable an implementation of advanced redirected walking algorithms [170], allowing users to navigate a point cloud in the same way they would explore a real-world scenery. In this context, redirected walking in would be especially important as the discrepancy between the available physical space and the area covered by the scan tends to be significant.

[8] https://www.tobiipro.com/product-listing/vr-integration/

[9] https://pupil-labs.com/blog/2018-04/htc-vive-pro-eye-tracking-add-on/

[10] https://www.oculus.com/blog/half-dome-updates-frl-explores-more-comfortable-compact-vr-prototypes-for-work/

Chapter 5

Web-Based Rendering of Enriched Point Clouds

In this chapter, the web-based rendering of enriched point clouds is discussed in depth, providing examples for server-side as well as client-side renderers and describing the specific advantages and disadvantages of those strategies. The chapter is based in parts on the author's scientific publications in [1] and [4].

The following sections are structured as follows: Section 5.1 further motivates this chapter and gives a more detailed introduction. Requirements deemed crucial for a system enabling the web-based visualization and collaborative exploration of enriched point clouds are defined in Section 5.2, followed by a proposed system architecture fulfilling these requirements. Section 5.3 focuses on the concrete implementation of the web-based rendering, whereas Section 5.4 follows up with an in-depth performance evaluation. Section 5.5 gives conclusions and outlines future research directions.

5.1 Introduction

The rendering approaches presented and evaluated in Chapters 3 and 4 can be applied to arbitrary large data sets by spatially subdividing these into small, representative subsets that are selected for real-time rendering at runtime based on different criteria such as memory budgets, targeted frame rates, and view setups. However, these rendering approaches were primarily designed with individual users operating on dedicated hardware in mind and, thus, come with several drawbacks:

- They assume a direct access to the corresponding point cloud, which generally restricts their application to systems with massive local storage capacities. First and foremost, this affects mobile devices, which are steadily replacing stationary computers as primary working devices as well as sources of entertainment. However, even stationary systems may struggle with providing sufficient local disk space, considering constant advances in data precision and density achieved by state-of-the art sensing hardware.
- Enriched point clouds are rarely inspected in solitude. Instead, exploration tasks —especially in the context of large data sets— are often conducted collaboratively, necessitating interaction techniques that are synchronized between the devices of all

involved stakeholders. For example, annotations and measurements created by one user may also be of interest for other users and, thus, should be shareable across devices.

- With point clouds being established as a universal data category for a large number of geospatial and non-geospatial applications, there is a growing demand for the seamless integration of processing as well as rendering functions for enriched point clouds into existing workflows and systems. In particular, this necessitates the application of service-oriented and web-based design principles that can leverage distributed and cloud-based hardware setups (see Chapter 2). In such setups, however, device-specific, independent copies of the data —as they are assumed by the previously discussed rendering techniques— are prone to generate data inconsistencies and synchronization conflicts.

To address these drawbacks, web-based rendering approaches are required that limit workload and data traffic on client-side by using a central server infrastructure to maintain and distribute the data. Rendering directly on the server and only transferring the rendered images to the client is commonly referred to as a *thin client* approach [55, 76]. Alternatively, *thick client* approaches can be applied that delegate the rendering to the client side — in that case, the server is only responsible for selecting and transferring the data to the client [86, 158].

In this chapter, a web-based system for the interactive and collaborative exploration and inspection of enriched point clouds is presented that provides functionality for both, thin-client and thick-client applications and, thus, scales for client devices that are vastly different in computing capabilities. The system is based on standard WebGL on the client side and can render point clouds with billions of points. It uses spatial data structures and LoD representations to manage the point cloud data and to deploy out-of-core and web-based rendering concepts. Different point-based rendering techniques and post-processing effects are provided to enable task-specific and data-specific filtering and highlighting, e.g., based on per-point surface categories or temporal information. A set of interaction techniques allows users to collaboratively work with the data, e.g., by measuring distances and areas, by annotating, or by selecting and extracting data subsets. Additional value is provided by the system's ability to display additional, context-providing geodata alongside point clouds (Figure 5.1) and to integrate task-specific processing and analysis operations. The presented techniques and the prototype system are evaluated with different data sets from aerial, mobile, and terrestrial acquisition campaigns with up to 120 billion points to show their practicality and feasibility.

5.2 Requirements and Concepts

The following requirements were identified as crucial for a system enabling the web-based visualization and collaborative exploration of enriched point clouds and 4D point clouds:

Figure 5.1: *Example of a point cloud rendered with the web-based rendering system presented in this chapter. Context-providing geodata such as 2D maps and 3D terrain models can be seamlessly integrated into the visualization.*

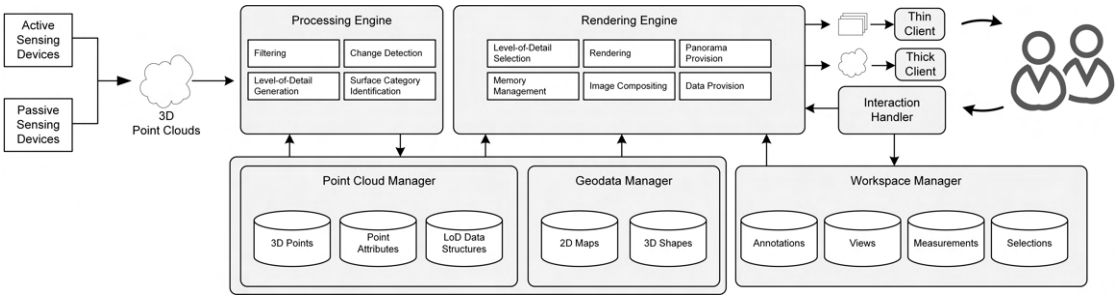


Figure 5.2: *System architecture showing data flow between integration, processing, visualization, and interaction components.*

R1 Use of point clouds as a fundamental geometry type instead of generalized mesh-based representations to enable a direct and unfiltered provision of the data.

R2 No limitations regarding used acquisition methods as well as density, resolution, and scale of the data (e.g., hundreds of billions of points, complete countries).

R3 Support for varying hardware platforms and computation capabilities, ranging from high-end desktop computers to low-end mobile devices.

R4 Distributed data storage to enable load balancing and to adjust for data specific requirements (e.g., certain point clouds might have to be stored on a specific server).

R5 Capabilities to prepare and clean up point clouds for the visualization (e.g., noise and outlier removal).

R6 Capabilities to conduct task and data specific analyses on point clouds (e.g., surface category extraction) to provide adaptive and task specific content.

R7 Visualization of analysis results (e.g., surface categories) to enable task specific highlighting and filtering.

R8 Capabilities to compare and show differences between point clouds of the same site captured at different points in time (i.e., change detection).

R9 Integration of supplementary, context-providing geodata such as 2D maps.

R10 Provision of interaction techniques to inspect (e.g., measuring of distances, areas, volumes) and annotate point clouds.

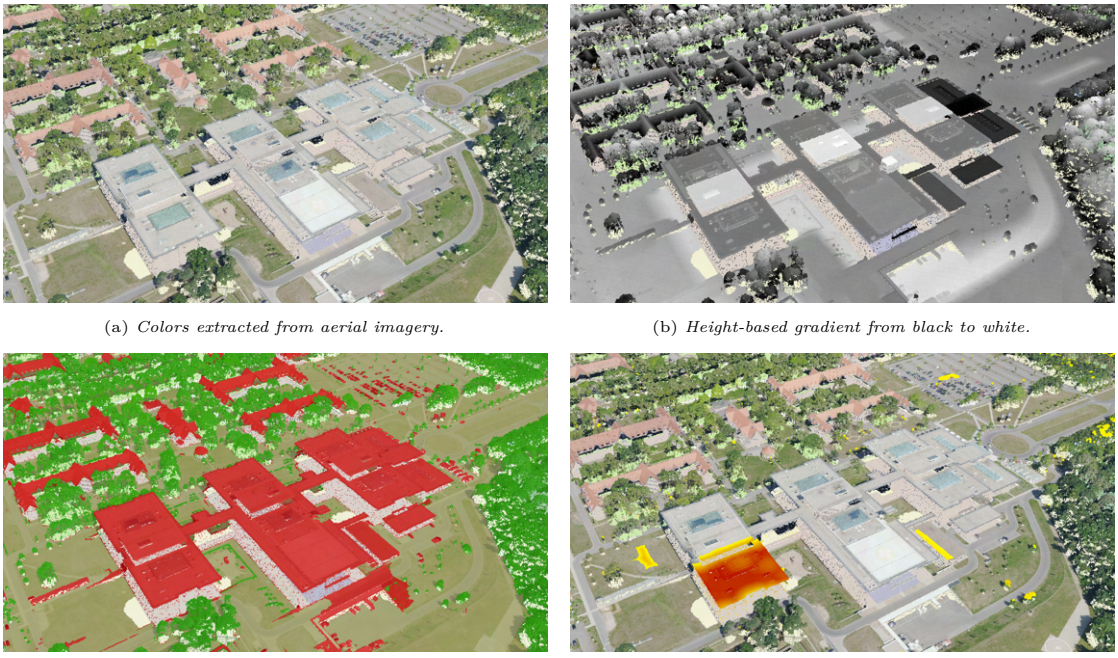
R11 Basic user management to customize data access.

R12 Capabilities to share specific rendering configurations, annotations and measurements with others (e.g., via link).

Based on these requirements a web-based system was designed and implemented that seamlessly combines functionality to integrate, process, and collaboratively explore enriched point clouds as well as supplementary, context-providing geodata. The proposed system (Figure 5.2) consists of the following major components:

Point Cloud Manager

Point clouds are organized in a single, homogeneous spatial data model. Access to that model is handled by the *point cloud manager* storing spatial information and additionally provided or computed per-point attributes (e.g., temporal information or surface categories) **(R1)**. LoD representations [69, 59] are required to efficiently access arbitrary data subsets of any size based on spatial, temporal or any other data layer. These representations as well as additional data layers can be generated by the *processing engine* **(R2)**. While the point cloud manager logically acts as a singular component, the



(a) *Colors extracted from aerial imagery.*

(b) *Height-based gradient from black to white.*

(c) *Colorization based on surface categories, i.e., green for vegetation, red for buildings, and brown for ground.*

(d) *Colors from aerial imagery combined with change detection results, i.e., gradient from yellow to red indicates the degree of change.*

Figure 5.3: *Different point-based rendering styles can be selected and configured at runtime.*

data itself may be stored in a distributed infrastructure, e.g., to maximize data throughput and network transfer rates or to account for data specific requirements regarding server location and data security **(R4)**.

Workspace Manager

The *workspace manager* handles information specific to a workspace, i.e., each user's private view of a specific data subset containing custom selections, measurements, annotations, view positions, and angles. Per default, each user operates in its own private workspace rather than sharing one globally with everyone else to avoid conflicting modifications **(R11)**. However, a given workspace may be shared via links **(R12)**. Each user may also own multiple workspaces.

Geodata Manager

The term *application-specific geodata* refers to additional geodata that should be used and rendered in combination with a point cloud to provide application-specific information layers **(R9)**. Examples are digital terrain models, aerial images, BIM models, or 3D city models. Like point clouds, these data types require supplemental LoD representations to allow for an interactive visualization. Application-specific geodata can be stored and provided by independent geospatial databases or geodata services, access to which is handled by the *geodata manager*.

Processing Engine

The processing engine conducts task and data specific operations on a given data subset. These operations range from (a) essential preprocessing steps (e.g., converting input data sets into a homogeneous georeference system or generating LoD representations), over (b) simple point cloud filtering (e.g., noise and outlier removal **(R5)**) to (c) more complex analyses (e.g., surface category extraction and change detection **(R6)**) deriving additional data layers. The operations can be accessed via web processing services implemented as separate web services that are individually combined and scheduled by the processing engine. Thus, existing web processing services for 3D point clouds can be easily integrated into the system. The results of each operation are automatically stored by the *point cloud manager* and can be seamlessly integrated by the *rendering engine* into depictions of the corresponding site **(R7)**. Behind the scenes, the processing engine uses a modular pipeline architecture combining parallel computing and out-of-core concepts whose details are described in Chapter 2.

Rendering Engine

Providing the core functionality of the presented system, the rendering engine is responsible for interactively visualizing three types of data: (a) Enriched point clouds featuring a varying number of data layers, (b) task-specific geodata providing context (e.g., maps **(R9)**), and (c) workspace elements resulting from user interactions (e.g., annotations or selection and measurement indicators **(R10)**). For each of those data types the corresponding manager is queried, retrieving only data subsets that are relevant for the current view and task. To highlight certain aspects of the data (e.g., temporal changes or surface categories in an area), different point-based rendering techniques and post processing effects can be combined **(R8)**. Changes to the currently applied render configuration can be made dynamically via the *interaction handler*) (Figure 5.3). In general, retrieved data subsets will be transferred to and rendered on client side, which minimizes the workload on the server (i.e., thick clients). As an alternative, server-side rendering can be applied to reduce the performance impact for clients (i.e., thin clients). Thus, the system scales for a broad range of devices, ranging from high-end workstations to mobile devices **(R3)**.

Interaction Handler

The interaction handler is responsible for handling user interactions as well as for updating the rendered data and workspace elements accordingly **(R10)** (Figure 5.4). Users may

- define or load workspaces,
- select which data subsets to render,
- configure the presentation of the data (with regards to applied rendering techniques),
- select, query, and highlight individual points or groups of points,

(a) *Measuring of distances between points.*

(b) *Measuring of areas defined by multiple points.*

(c) *Annotation of selected points or areas.*

(d) *Selecting areas of interest.*

Figure 5.4: *Overview of implemented interaction techniques.*

- measure distances and areas between selected points,
- annotate selected points or areas,
- modify annotations,
- saving and loading view positions and angles.

5.3 Rendering Engine Implementation

To seamlessly combine point clouds, context providing geodata and interactive workspace elements into a homogeneous visualization, a multi-pass rendering pipeline similar to the ones presented in Chapters 3 and 4 is used that consists of three distinct stages (Figure 5.5).

5.3.1 LoD and Data Subset Selection

While point clouds may easily contain billions of points, only a fraction of that data is required to render a frame. Subsets of the point cloud that are manageable by available CPU and GPU capabilities can be queried dynamically from the point cloud manager by specifying the current view frustum, main and GPU memory budgets as well as task specific qualifiers (e.g., value ranges for selected per-point attributes) to filter the corresponding data sets. In particular, the resulting *screen-space error* is used as a metric to evaluate potentially fitting subsets, optimizing towards a *maximum allowed*

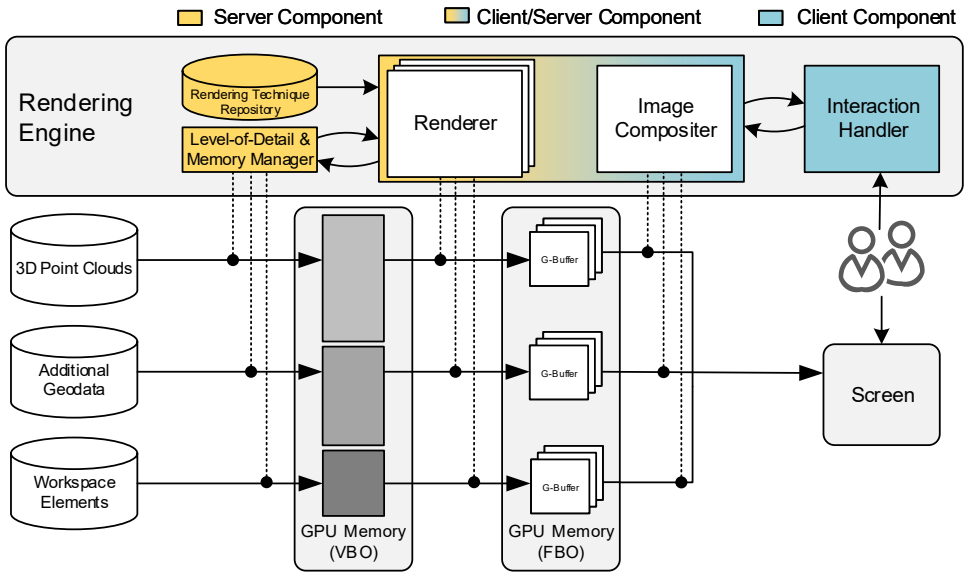
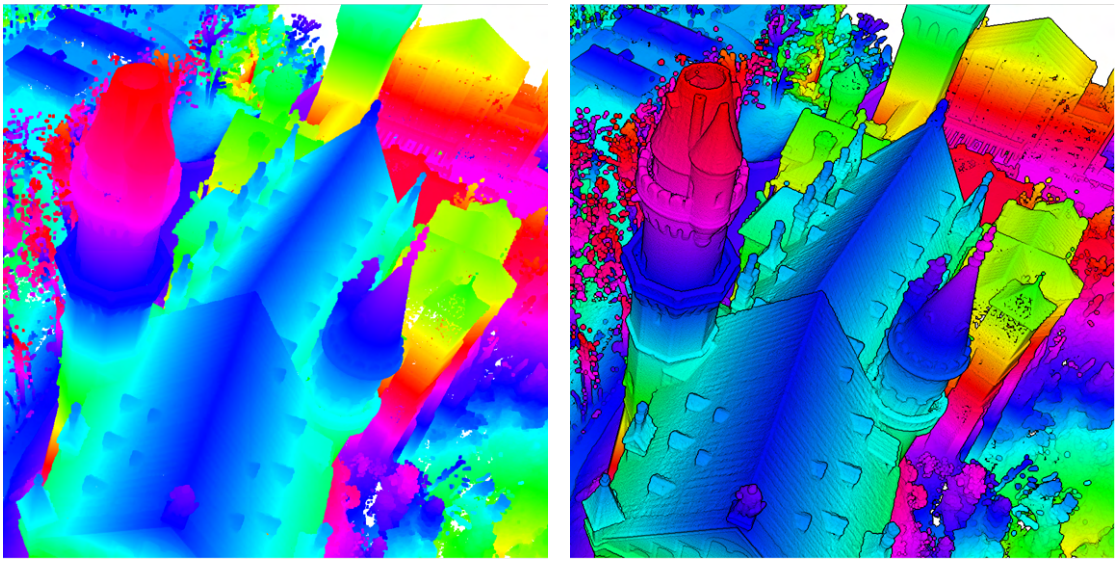


Figure 5.5: *Overview of the rendering pipeline. Each data type is managed and rendered separately.*

screen-space error: The higher the allowed screen-space error, the less points need to be queried and rendered, albeit at the cost of further reducing precision and density of the point cloud depiction. To accommodate for changing network latencies, the maximum allowed screen-space error can be adjusted at runtime. Since hardware specifications of clients cannot be queried from a web browser, main and GPU memory budgets need to be specified manually. To assist users in specifying a reasonable budget, a set of predefined budget configurations that have been evaluated for a variety of common hardware setups is optionally provided.

To enable an efficient subset retrieval, the data is hierarchically subdivided in a pre-processing step before being made accessible by the point cloud manager. Similarly to Chapters 3 and 4, a multi-layered data structure is constructed: For each data set, a separate spatial data structure is generated that best compliments the spatial distribution of the corresponding points (e.g., quadtrees for airborne data sets, octrees, or kd-trees for terrestrial data sets, see Section 2.4). In turn, those spatial data structures are integrated into an overarching quadtree, allowing to efficiently answer queries stretching across multiple data sets. Compared to uniform, single-layer spatial data structures, e.g., as they are used by Potree [162], this avoids a time-consuming rebalancing when new point clouds are added while simultaneously ensuring balanced tree structures and minimal data access times. Context providing geodata and workspace elements are handled similarly and can be queried simultaneously from their respective manager when required.



(a) *Colorization using a height gradient.* (b) *Same color scheme combined with EDL*

Figure 5.6: *Post-processing effects such as EDL facilitate visual filtering and highlighting.*

5.3.2 Rendering and Image Compositing

After being queried from the respective managers, point clouds, context providing geodata and interactive workspace elements are rendered into separate G-Buffers combining 2D textures for, e.g., color, depth, normal, or ID values. The use of ID values is important to separate point clouds from context data. Each rendered point has a unique identifier, stored in an ID texture, to allow for an efficient point selection, e.g., to implement interaction features. In addition, different rendering styles can be configured and applied at runtime. As an example, size and color of each point can be modified based on selected data layers (e.g., surface categories, geometrical information) to enable task specific visual filtering and highlighting (Figure 5.3). Similarly, several options exist to dynamically adjust the appearance of mesh-based geometry, ranging from transparency settings to changeable texture mappings.

A final image compositing stage is used to merge the separate G-Buffers, i.e., to combine several independently generated views of point sub-clouds into a final image. For example, image-based post processing effects emphasizing edges and depth differences (e.g., SSAO or EDL) can be applied at that stage to improve the visual identification of structures within point cloud depictions (Figure 5.6). As described in detail in Section 3.3.3, the ID textures stored by the G-Buffers provide efficient means to select individual points in real-time, which is an essential requirement to support annotating points or measuring distances and areas.

5.3.3 Web-based Rendering

To accommodate for client devices with varying computation capabilities, different web-based rendering concepts are combined with the presented rendering pipeline (Figure 5.7).

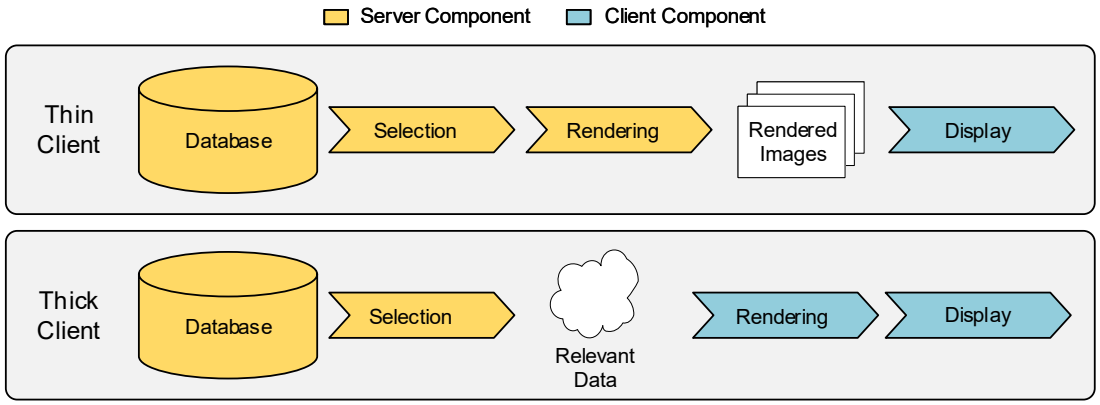
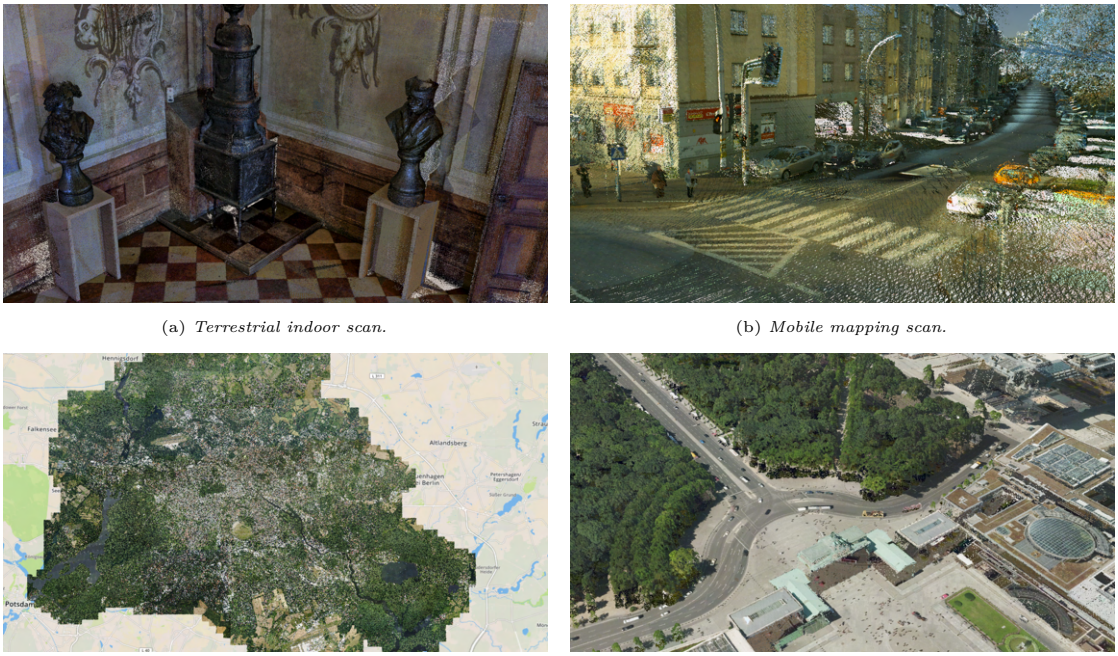


Figure 5.7: *Comparison of web-based rendering concepts: Thin clients vs thick clients*

A thick client application is provided that uses a central server infrastructure to organize, process, select, and distribute the data, but delegates the actual rendering of selected data subsets to the clients. This approach significantly reduces workload on server side, allowing to serve massive numbers of clients simultaneously. Transferred data subsets are cached on client side up to a device specific limit, thus, minimizing the frequency of data requests for subsequent frames. In fact, additional data subsets are only required if the view frustum changes significantly, whereas inspecting the transferred data or changing the applied rendering style triggers no such requests.

Alternatively, in the sense of a thin client approach, the data can be rendered directly on the server, supplying only the resulting images. While this comes with the drawback of increased workload on server side as any user interactions trigger a new data request, hardware requirements for clients are notably reduced. A common optimization for such thin client applications is to render and transfer cube-maps or virtual panoramas instead of individual images [55, 76]. This provides clients with efficient means to locally reconstruct the 3D scene for a specific view position. Thus, the data only must be rendered anew whenever the view center or the rendering style are modified, which significantly reduces the frequency of data requests for subsequent frames. Our system provides a thin client application that expands that concept, distributing not only traditional 2D panoramas but also stereoscopic panoramas. Thus, emerging virtual reality technologies allowing for an immersive exploration of point clouds even on mobile devices can be easily integrated. We generate those stereoscopic panoramas by rendering several equally sized image strips along a *viewing circle* that are stitched together in a post-processing step [121]. The visual quality of the panoramas depends on the requested resolution as well as the number of image strips; both settings can be specified upon requesting a new panorama. To further reduce overall network load, both applications dynamically compress and decompress the transferred data, using common standards such as gzip (for thick clients) and png (for thin clients), respectively. To maximize visual quality, lossy compression standards (e.g., jpeg compression) are not applied.



(a) *Terrestrial indoor scan.*

(b) *Mobile mapping scan.*

(c) *Airborne scan of a city (zoomed out).*

(d) *Airborne scan of a city (zoomed in).*

Figure 5.8: *Scenes used during the performance tests.*

5.4 Performance Evaluation

The presented concepts have been implemented on the basis of several C++ and Javascript libraries. The processing engine uses *CUDA* [1] and the *Point Cloud Library* [2]. Regarding the rendering engine, WebGL and *CesiumJS* [3] provide the technological basis for thick client applications. For thin client applications, server-side rendering is based on OpenGL, *glbinding* [4] and *GLFW* [5]. On client-side, *Three.js* [6], *WebGL*, and *WebVR Polyfill* [7] are combined to display 2D as well as stereoscopic panoramas. Data compression is implemented based on *gzip* [8] and *lodePNG* [9], respectively. Evaluated point clouds are represented by separate kd-trees, that in turn are integrated into an overarching quadtree. The decision in favor of kd-trees was made to optimize the balancedness of the tree structures, speeding up the subset retrieval, albeit at the cost of a prolonged preprocessing. The spatial data structures and corresponding data subsets are serialized into files acting as a point cloud database. Similar, file-based approaches are applied to store and access context-providing geodata and workspace elements.

[1] https://developer.nvidia.com/cuda-zone
[2] http://pointclouds.org
[3] https://cesiumjs.org
[4] https://github.com/cginternals/glbinding
[5] http://www.glfw.org
[6] https://threejs.org
[7] https://github.com/immersive-web/webvr-polyfill
[8] http://www.gzip.org
[9] http://lodev.org/lodepng/

Table 5.1: *Average data throughput of the processing engine.*

Processing Operation	Average Data Throughput
Noise & Outlier Filtering	1.26B pts/hour
Surface Category Identification	0.10B pts/hour
Change Detection	1.42B pts/hour
Kd-Tree Generation	4.85B pts/hour

5.4.1 Test Setup and Results

A desktop computer featuring an AMD Ryzen 7 1700 CPU, 32 GB main memory and an NVIDIA GeForce GTX 1070 with 8 GB device memory was used as a server for the performance tests. The test data sets included a terrestrial, indoor scan of an individual site (1.33 billion points), a mobile mapping scan (2.57 billion points) and a massive, multi-temporal data set of an urban region (120 billion points) captured by airborne devices. For all data sets essential preprocessing steps (i.e., spatial data structure generation) and filtering (i.e., noise and outlier removal) were performed by the processing engine. In addition, surface categories (i.e., ground, building, vegetation) and changes in comparison to earlier scans were identified for the airborne data set, allowing to evaluate the system's ability to dynamically combine different rendering styles. The average data throughput for the applied processing operations is listed in Table 5.1.

The rendering engine was evaluated based on four different scenes (Figure 5.8) with client applications running on a number of different devices and web browsers (Tables 5.2 and 5.3). As opposed to aforementioned state-of-the-art web-based rendering frameworks for 3D point clouds such as Potree, Plasio, GVLiDAR or ViLMA, the rendering engine presented in this chapter provides both, a thick and a thin client renderer. Thus, the rendering process can be shifted dynamically between client and server side depending on network conditions and a client's computing and graphics capabilities, allowing to support a broader range of hardware platforms.

The presented CesiumJS-based thick client implementation allows to render several millions of points simultaneously at interactive frame rates (i.e., >30 fps) on standard desktop computers and notebooks (Table 5.6). On mobile devices, frame rates are significantly lower due to the more limited computing capabilities. However, arbitrary large data sets can be visualized on all evaluated devices by assigning device-specific memory budgets, thus, limiting the density of the point cloud depiction. Overall, rendering performance and visual quality are similar to what can be achieved by Potree, Plasio, GVLiDAR or ViLMA. However, the presented thick client implementation allows to seamlessly integrate additional geodata as well as analysis results which facilitates an in-depth inspection.

On the other hand, the presented thin client implementation provides a uniform rendering quality on all client devices since the panoramas are generated on server side, minimizing workload on client side. On all evaluated devices measured frame rates stayed

Table 5.2: *Devices used to evaluate the rendering engine. All web browsers were updated to the latest version as of 04/20/2018.*

Client Device	CPU	GPU
Lenovo M710t	Intel Core i7-6700	GeForce GTX 1050Ti
Macbook Pro 13”	Intel Core i5-4278U	Intel Iris 5100
iPhone SE	Apple A9 @ 1.84 GHz	PowerVR GT7600
Galaxy s7	Samsung Exynos 8890	ARM Mali-T880 MP12

Table 5.3: *Devices used to evaluate the rendering engine. All web browsers were updated to the latest version as of 04/20/2018.*

Client Device	Main Memory	Evaluated Web Browsers
Lenovo M710t	32GB	Chrome, Firefox, Opera, Edge
Macbook Pro 13”	16GB	Safari, Chrome, Firefox
iPhone SE	2GB	Safari Mobile, Chrome Mobile
Galaxy s7	4GB	Samsung Internet Browser, Chrome Mobile

close to the corresponding display’s refresh rate (e.g., 60 fps on the Galaxy S7), making the approach applicable to state-of-the-art VR devices such as GearVR or Oculus Rift. The performance of the panorama generation is primarily influenced by the requested resolution and to a lesser degree on the number of image strips used (Table 5.7).

For all evaluated scenes, thick client applications require to transfer significantly more data for an individual scene than thin clients as long as no reusable data subsets have been cached from previous requests, even if gzip compression is applied (Tables 5.4 and 5.5). However, they do not require all those data subsets at once, allowing to update the scene progressively. Furthermore, while exploring a point cloud, the view will usually change only gradually across subsequent frames, allowing for thick clients to reuse many of the previously transferred data subsets, thus, resulting in smaller and faster scene

Table 5.4: *Average amount of data transferred by the rendering engine based on the scenes defined in Figure 5.8. For thin clients, a stereoscopic panorama must be newly created per request. While the same, device-dependent resolution was requested for each scene, different entropies affected the compressed image size.*

Scene	Thick Client	Thin Client
Terrestrial	156.2 MB	4.68 MB
Mobile Mapping	140.7 MB	4.16 MB
Airborne (zoomed out)	16.1 MB	4.15 MB
Airborne (zoomed in)	82.4 MB	4.54 MB

Table 5.5: *Average data transfer times of the rendering engine based on the scenes defined in Figure 5.8. For thin clients, a stereoscopic panorama must be newly created per request. For thick clients, no additional calculations are necessary.*

Scene	Thick Client	Thin Client	
		Panorama Generation Time	Transfer Time
Terrestrial	16.18s	5.27s	1.36s
Mobile Mapping	14.15s	5.05s	1.27s
Airborne (zoomed out)	3.43s	4.96s	1.22s
Airborne (zoomed in)	8.09s	5.13s	1.32s

Table 5.6: *Average performance rate of the thick client for different point budgets based on the airborne data set (Figure 5.8 (d)).*

Number of Points	Lenovo M710t	Macbook Pro 13”	iPhone SE	Galaxy s7
2M pts	122.63fps	53.85fps	41.83fps	39.96fps
4M pts	84.48fps	45.63fps	36.44fps	35.29fps
6M pts	63.23fps	39.08fps	26.36fps	24.83fps
8M pts	56.87fps	35.83fps	19.65fps	18.43fps

updates over prolonged explorations. Changes to the rendering style as well interaction techniques such as picking, selecting, or measuring do not trigger any additional data requests at all and can be applied even under unstable network conditions. For thin client applications on the other hand, no parts of the previously transferred data can be reused if the currently used panorama becomes invalid: Navigating —apart from merely looking around from a fixed position— as well as rendering style adjustments require the server to generate and transfer a new panorama as a replacement. Similar to thick clients however, picking, selecting, or measuring can be conducted on the already transferred data and does not trigger any new data requests.

Table 5.7: *Panorama generation time for different configurations based on the terrestrial data set (Figure 5.8 (a)).*

Resolution	90 image strips	120 image strips	160 image strips
2360x1600 px	1.88s	2.26s	2.29s
2360x3200 px	4.20s	4.68s	5.27s
2360x6400 px	6.17s	7.14s	7.88s

Expandability

The proposed web-based system can be easily adapted for specific applications by adding custom visualization techniques or algorithms for data analysis: Per-point attributes as well as the pipeline nodes used by the processing engine share common interfaces which facilitates the implementation of additional *importers*, *exporters*, or *tasks*. The corresponding *pipeline plans* are defined via JSON files and can thus be easily customized. Similarly, the rendering engine allows to define and apply custom GLSL shaders to adapt the visualization. Some examples of such expansions are presented in Chapter 6.

5.5 Conclusions and Future Work

Web-based visualization and exploration of enriched point clouds from aerial, mobile, or terrestrial data acquisitions represent a key feature for today's and future systems and applications dealing with digital twins of our physical environment. The presented web-based rendering approach scalably visualizes point clouds onto web-based client devices. To cope with enormous numbers of points, the implementation relies on spatial data structures and LoD representations, combined with different out-of-core rendering and web-based rendering concepts. Since the rendering process can be shifted from client side to server side, the system can be easily adapted to varying network conditions and to clients with a broad range of computing and graphics capabilities. Performance tests on data sets with up to 120 billion points show the usability of the system and the feasibility of the approach. This is further underlined by the case study presented in Chapter 6.

As future work additional case studies should be conducted regarding to the system's performance in distributed server environments. Various rendering techniques allow to filter and highlight subsets of the data based on any available per-point attributes (e.g., surface categories or temporal information), which is required to build task-specific or application-specific tools. Various interaction methods (e.g., for collaborative measurements and annotations), built-in support to display context-providing, mesh-based geodata, and the possibility to conduct different processing and analysis operations provide additional features. The system could be further extended by integrating additional analyses (e.g., for asset detection, or surface segmentation) [172, 81] as well as by specialized interaction techniques. For example, Scheiblauer and Wimmer [157] and Wand *et al.* [181] propose spatial data structures that allow for an interactive editing of 3D point clouds. In addition, sophisticated visualization techniques for 4D point clouds are becoming increasingly important to understand captured environments [138] and, thus, should be further investigated.

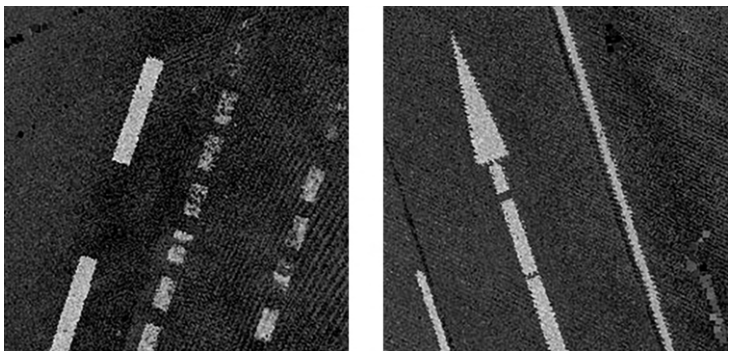
Chapter 6

Case Studies and Applications

The presented rendering and interaction techniques provide the technological foundation for specialized tools and applications addressing specific real-world use cases. Some of these tools and applications are still in a conceptual stage, depending on a more widespread adoption and acceptance of underlying technology by the general public before they can be evaluated in practice on a large scale. As an example, consumer and enterprise level VR setups, as used in Chapter 4, are still widely considered as niche products that require significant space and financial investment, even though market sizes are predicted to grow significantly over the next years [17]. In other cases, visualization merely serves to facilitate subsequent analysis steps. One such example is described in [11], where different point-based rendering techniques are used to generate highly detailed screenshots of point cloud depictions that can be utilized by established convolutional neural networks for image segmentation to identify surface categories that are mostly two-dimensional by nature, such as road markings (Figure 6.1).

As opposed to that, this chapter focuses on systems and applications that (a) use interactive depictions of point clouds as a core feature and (b) can be practically deployed and evaluated on a large scale right away. In the following, two case studies are presented that showcase the practicability and robustness of such applications in more detail.

(a) *Point cloud with detected road markings represented as orange shapes rendered on top of the point cloud.*



(b) *Examples of rendered images from a point cloud, showing different types of road markings. Intensity values are represented in grayscale, lighter colors have higher intensity values.*

Figure 6.1: *Point-based rendering techniques play an important role in image-based classification approaches for point clouds as discussed in [11].*

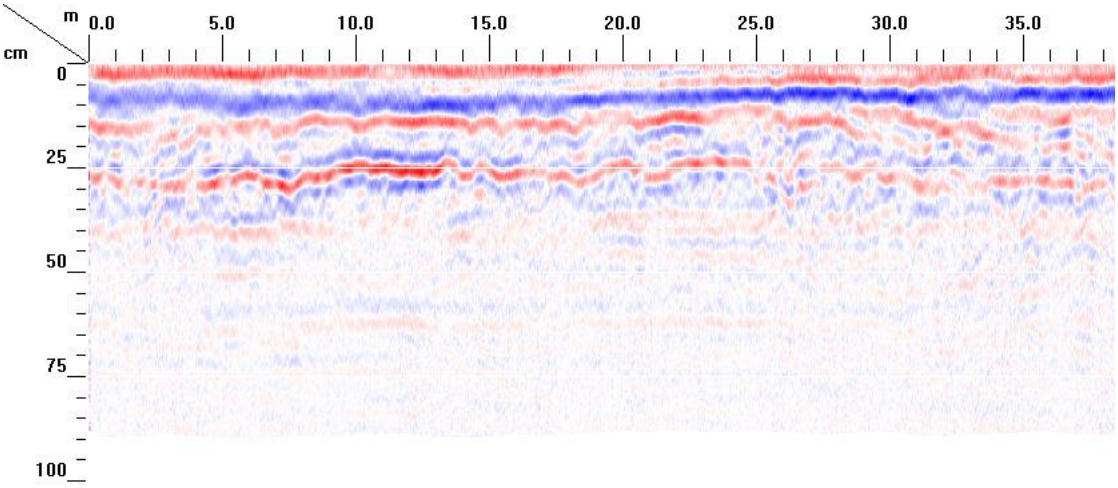


Figure 6.2: *Typical visualization of a GPR B-scan, with the x-axis representing the traveled distance and the y-axis representing the results of the individual A-scans. Using a two-color representation, the direction of the signal's amplitude is represented by hue and its size by saturation.*

6.1 Combined Visual Exploration of GPR Data and Point Clouds for Road Environments

This section is based in parts on the author's scientific publications in [10] and [9].

While a lot of semantic information and insights can be derived directly from the raw point data, other types of geodata, that often can be automatically captured alongside the point cloud, may still provide crucial additional information for certain use cases. One such example are ground penetrating 2D radar scans.

Ground penetrating radar (GPR) has been used for below-ground analyses for several decades, allowing to measure material properties several meters below ground and to create insights about the non-visible foundation of roads and pathways [46]. GPR scanners operate by emitting electromagnetic waves into the ground and receiving the reflected signal from pavement and soil. Since the ground's structure impacts propagation of the emitted signal, the reflected signal provides information about the materials' condition. Just as sensors for mobile mapping scans, GPR scanners are usually mounted on scanning vehicles which can drive unintrusively along with traffic, adding an additional data source for the region not accessible by LiDAR scanning [129]. Once captured, the radar scan data is usually analyzed in the form of so-called *B-scans*, i.e., consecutive sequences of individual measurements (i.e., so-called *A-scans*) taken alongside the driving trajectory of the scanning vehicle. A typical visualization of a B-scan is shown in Figure 6.2.

The combination of above-ground point clouds and below ground GPR data enables a more extensive analysis of road environments by using two combined data sources instead of evaluating each one separately. As an example, a common use case for GPR data inspection is to find certain areas with an increased chance of developing pot-holes [79].

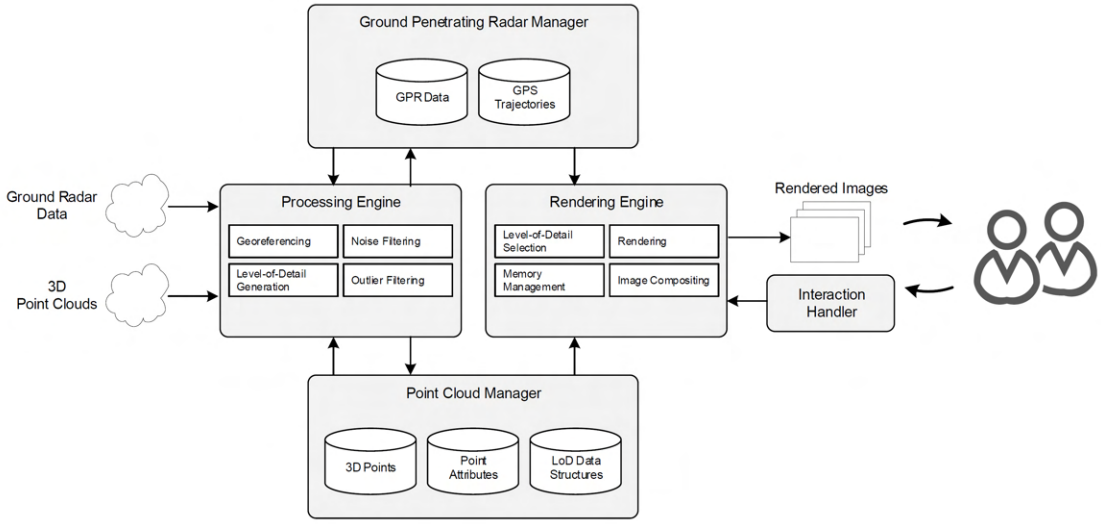


Figure 6.3: *System architecture showing data flow between integration, processing, visualization, and interaction components.*

By adding road surface information from a point cloud of the same area, false positives like manholes can easily be distinguished from other anomalies in the road's subsoil. This section presents desktop-based system for the combined interactive exploration of point clouds and GPR data. In particular, the following requirements were identified as crucial for the rendering system's intended purpose:

R1 No limitations regarding used acquisition methods, as well as the number, scale and size of the data sets. The latter is especially important for point clouds, as those can easily contain hundreds of gigabytes of raw data.

R2 Correct positioning of GPR data and point clouds into a homogeneous spatial reference system.

R3 Occlusion-free visualization of individual B-scans within a GPR data set.

R4 Visual filtering and highlighting techniques to enable a focused inspection of areas of interest, that can be defined at runtime.

6.1.1 System Overview

To address the aforementioned requirements, the rendering approach presented in Chapter 3 was extended and integrated into a simplified, *strictly offline* adaptation of the system architecture presented in Chapter 5. In particular, the rendering system consists of the following components (Figure 6.3):

- **Point Cloud Manager.** All point clouds are sorted into a single, homogeneous spatial data model. Access to that model is handled by the point cloud manager,

storing spatial information together with additional per-point attributes (e.g., color values). LOD representations, which are essential for an efficient rendering of massive data sets (Chapter 3), are generated by the *processing engine* **(R1)**.

- **GPR Manager.** Fulfilling a similar role as the point cloud manager, the *ground penetrating radar manager* enables efficient access to GPR data, which is stored in combination with simultaneously captured GPS trajectories. Based on that information, GPR data can be positioned precisely within point clouds of the corresponding area **(R2)**. Individual B-scans within a GPR data set can be accessed separately **(R3)**. Since the size of the GPR data sets evaluated in the context of this chapter is neglectable by comparison (i.e., 25.7MB raw data per B-scan), no LoD representations were deemed necessary.

- **Processing Engine.** The processing engine conducts different pre-processing operations on given data sets, ranging from (a) georeferencing data (e.g., by combining GPR data and GPS trajectories), over (b) data cleaning (e.g., filtering of noise and outliers in 3D point clouds) to (c) generating LOD representations for point clouds. Using the modular pipeline architecture described in Chapter 2, the processing engine allows running and scheduling multiple operations in parallel. Results of those operations are automatically stored by the point cloud manager and ground penetrating radar manager, respectively.

- **Rendering Engine.** The *rendering engine* is responsible for providing an interactive, combined visualization of point clouds and GPR data. To that end, the multi-pass rendering approach presented in Chapter 5 is utilized, allowing to apply image-based post processing effects that facilitate visual filtering and highlighting. For each data type the corresponding manager is queried, retrieving only data subsets that are relevant for the current view and task. Changes to the current render configuration (e.g., regarding applied post processing effects) can be made at runtime via the *interaction handler*.

- **Interaction Handler.** The interaction handler updates the visualization according to user interactions. In particular, users can (1) change view position and angle, (2) select B-scans they want to focus on, (3) select and configure applied post-processing effects, and (4) define areas of interest for highlighting **(R4)**.

6.1.2 Visualization Techniques

The combined visualization of point clouds and GPR data is based on two major user interface components: A interactive 3D scene view and a 2D user interface.

3D Scene View

The first step towards integrating GPR data and point clouds into a single visualization is projecting each B-scan onto the captured GPS trajectory **(R2)**. To prevent different

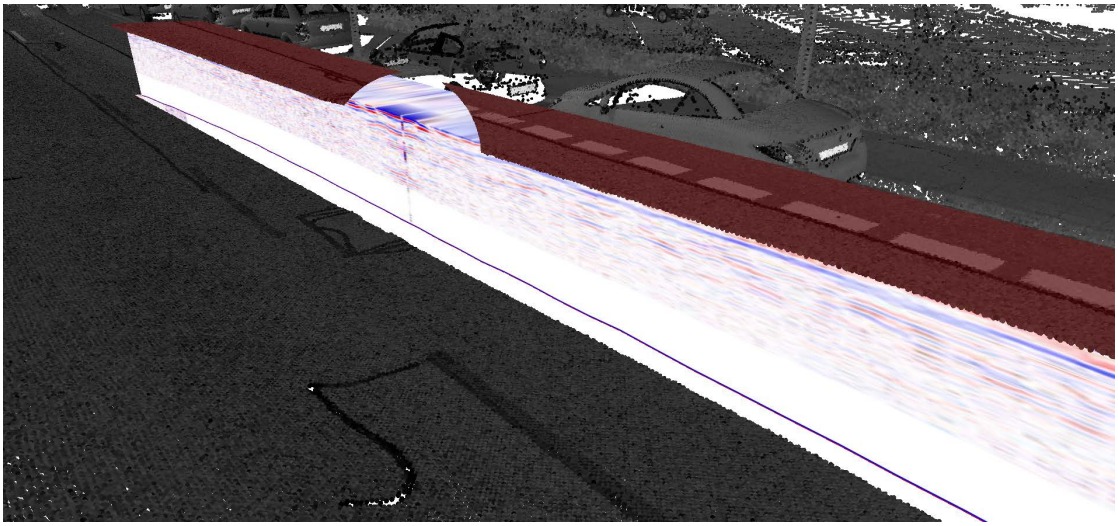


Figure 6.4: *Points under the cuboid are elevated and highlighted. An interactive lens shows the cuboid's surface below the points.*

B-scans from occluding each other, individual B-scans can be hidden dynamically **(R3)**. Each GPR data set is represented by a *cuboid*, covering the amount of space scanned by the GPR sensors, that is rendered onto the GPS trajectory (Figure 6.4). To fill the area in between the B-scans, their values are interpolated. A 3D texturing approach guarantees the possibility of slicing the cuboid –both vertically and horizontally– as well as moving it along the trajectory. That way, the cuboid can be restricted to specific areas of interest, thus, facilitating visual filtering and highlighting **(R4)**.

The visibility and usability of the cuboids is increased by raising them onto the corresponding GPS trajectory rather than leaving them below ground level **(R3)**. To keep the spatial context, the points located originally above the cuboid are translated alongside and highlighted for a better contrast to non-translated points (Figure 6.4). Furthermore, to enable a direct view onto the cuboid's surface, an interactive see-through lens similar to the one presented in Chapter 3 is provided that hides points around the mouse cursor. As an alternative, the see-through lens mode may be inverted, displaying only points around the cursor. This allows to focus on the GPR data, while keeping once more the spatial context **(R4)**.

2D User Interface

To facilitate an in-depth exploration of the GPR data, a supplementary widget is provided that shows all B-Scans in full length in 2D (Figure 6.5) and allows to configure the 3D scene view.

In particular, users may change how the cuboid of the given GPR data set is rendered. This includes (1) setting its elevation relative to ground level, (2) cropping to specific start and end points, (3) cropping to specific minimum and maximum radar scanning

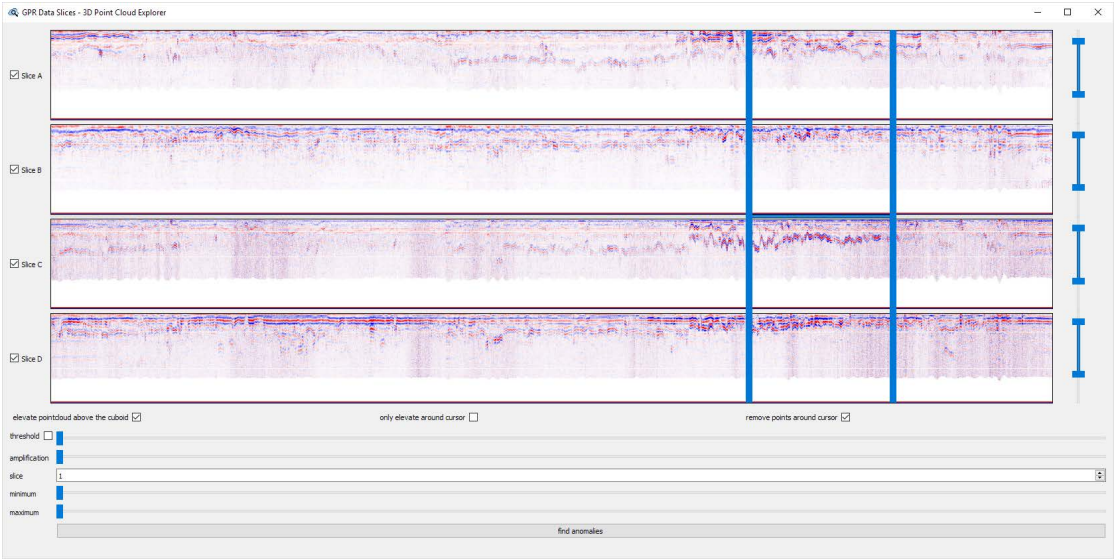


Figure 6.5: *2D User Interface for GPR data with cropping and thresholding options.*

heights, and (4) hiding specific B-scans altogether. Cropping to specific start and end points enables users to move both, the cuboid and the corresponding slice in the 2D view, back and forth. Doing so moves the camera position in the 3D view accordingly, ensuring that the view is always centered on the cuboid **(R4)**.

Furthermore, users may change, how textures are generated from a GPR data set. While the input data includes raw information about the reflected signal picked up by the receiver, the generated textures show the amplitudes of these values, with positive and negative values color coded in red and blue. Users may change how much these values should be amplified, since their range can vary greatly. As an example, a much stronger amplification should be applied when exploring parts of the data with small differentials. Furthermore, parts featuring drastic changes (i.e., most often points of interest) can be highlighted by specifying thresholds. As a consequence, these parameters facilitate identifying anomalies in specific regions of the GPR data.

6.1.3 Evaluation

Similar to the rendering system evaluated in Chapter 3, the presented adaptation was implemented based on C++, OpenGL, GLSL, and OpenSceneGraph. Test data consisted of four GPR B-scans continuously captured in driving direction alongside a 650m long trajectory, as well as a mobile mapping LiDAR scan of the same area. The four B-scans were captured in parallel, using radar antennas sensing with a frequency between 1000 and 2000 MHz. Reaching a depth between 0.45m and 0.90 m, each radar antenna captured the 650m of road data with a density of 13,146 data points, holding 512 4-byte samples each. The mobile mapping LiDAR scan featured a density of at least 1000 points per square meter in those areas covered by ground penetrating radar. In addition to spatial

information, the intensity of each returned laser ray was measured as well. In the absence of color information, these intensity values were visualized as grayscale values to enhance the point cloud depiction.

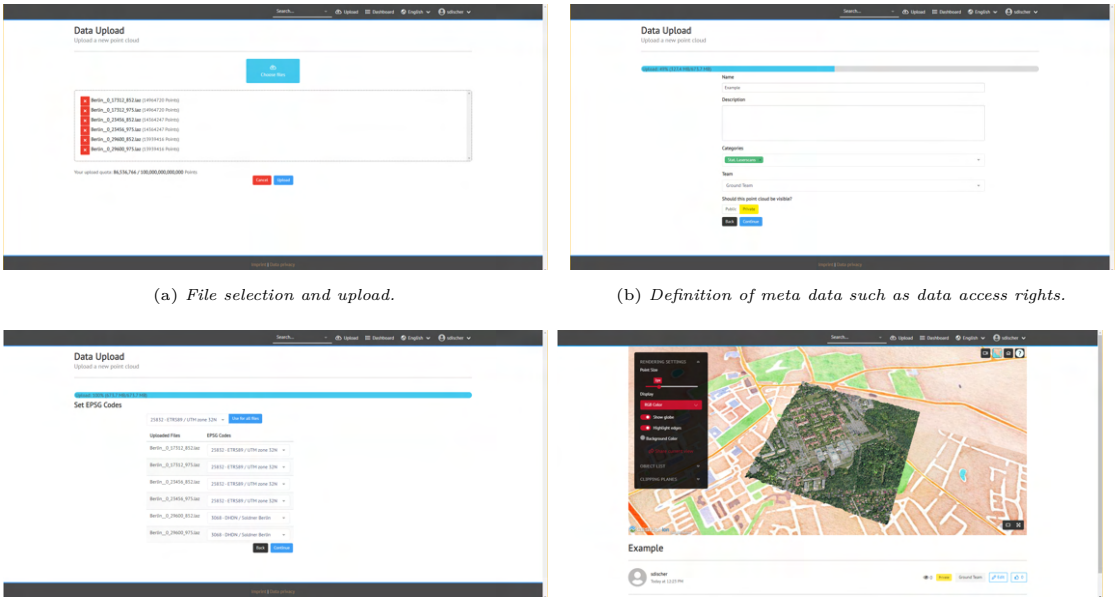
Usability

Having antennas operating at different frequencies —and therefore featuring different pickup patterns, maximum depths, and accuracies— is useful for having nearly the same region covered with two completely different settings. However, it makes visualization more challenging since neighboring B-scans are not directly comparable anymore. Thus, interpolating the 2D B-scans to create a 3D representation of the captured data might lead to unexpected results when B-scans with differing antenna settings are active. Also, B-scans may easily occlude each other. Since an important part of the visualization is that the coordinates of the radar scan are exactly mapped to the point cloud, adjusting the distance between the B-scans is not a viable option. Thus, users must hide unwanted B-scans to make otherwise occluded B-scans visible.

Performance

As discussed in Chapter 3, the presented rendering system allows generating interactive frame rates for arbitrary point clouds. Regarding ground penetrating radar, the supplied B-scans first must be loaded into OpenGL textures to generate the final rendered textures. Thus, the performance cost in this case is composed of the initial time to load the data, and the time at runtime to update and draw the B-scan textures. Since the few textures in the context of this case study feature a resolution of 13,146,512 pixels at a bit depth of four bytes for the raw data and one byte for result textures, and both updating and drawing of these textures is completely GPU-accelerated, this runtime cost is negligible in comparison to the one introduced by managing and rendering the point cloud. However, for bigger data sets, more advanced memory managing methods and LoD approaches might be needed to overcome main and GPU memory limitations.

As a conclusion, this case study exemplifies how the interactive rendering techniques, that have been presented throughout this thesis, can be applied in real-world scenarios to facilitate the design and implementation of domain specific interactive tools and applications. In addition to being applicable to 3D point clouds of arbitrary size and density, they offer many degrees of freedom for application design as they can be seamlessly combined with existing rendering techniques for other types of geometry such as 2D and 3D meshes or —as in this case— georeferenced image data. Some of the presented rendering techniques may even be directly applied to other types of geometry, requiring only minimal adaptation, as exemplified by the interactive see-through lenses used to provide unobstructed views of a cuboid's surface in areas of interest without losing spatial context.



(a) *File selection and upload.*

(b) *Definition of meta data such as data access rights.*

(c) *Definition of EPSG codes to georeference uploaded data sets.*

(d) *Visualization of the uploaded and configured data.*

Figure 6.6: *Web frontend used during the first case study allowing users to upload, prepare and explore enriched point clouds.*

6.2 Web-Based Management and Monitoring of Large-Scale Urban Development Projects

This section is based in parts on the author's scientific publications in [2], [1] and [6].

Following the ever-increasing availability and affordability of in-situ and remote sensing technology, enriched point clouds have been widely established as a central source of both, geospatial and non-geospatial information. A prime example of this are large-scale infrastructure and urban development projects. Here, point clouds are captured and updated numerous times throughout a project's lifetime to document and monitor current developments [167, 134]. As point clouds in this context may need to be collaboratively inspected, processed, and exported by several stakeholders of vastly different backgrounds [111], isolated visualization solutions that focus exclusively on specific use cases and individual users (e.g., Section 6.1) are often not applicable to such large-scale projects. Instead, sophisticated systems are required that combine web-based technology to store, update, analyze, and collaboratively inspect captured point clouds while also providing functionality to seamlessly integrate with existing workflows and external systems. One example of such a system was presented in depth in Chapter 5. This section follows up on the initial performance evaluation (Section 5.4) and provides several case studies that demonstrate the scalability of the presented system with regards to user base, data size, as well as available computing and graphics capacities in the context of large-scale infrastructure and urban development projects.

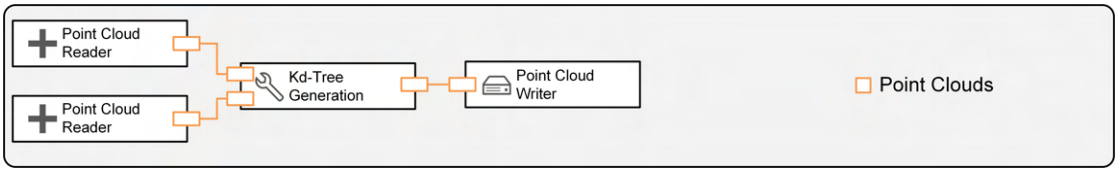


Figure 6.7: *Processing pipeline used for the first case study to generate kd-trees from a set of input data sets.*

6.2.1 First Case Study: Collaborative Interaction with Enriched Point Clouds

The *first case study* focused on the collaborative interaction with enriched point clouds in the context of a large-scale infrastructure project involving up to 10 concurrent users representing different stakeholders spread across Germany. The infrastructure project comprised several individual sites that were captured by air or —in the case of some especially relevant building complexes— via terrestrial scanning. In total, the scans amounted to 5.31 TB of raw data (E57 or las format) distributed across 144 individual data sets, each of which containing between 18 million to 4.1 billion points at an average point density of 6.1 points/m^2 (airborne scans) and 1.2 million points/m^2 (terrestrial scans), respectively.

Via a web frontend (Figure 6.6) users were able to (1) upload data sets asynchronously, (2) georeference them individually and (3) restrict data access to specific users. Simultaneously —given corresponding data access rights— users could collaboratively inspect and annotate point clouds that had already been added to the system. Rendering performance on client devices was consistent with the results presented in Section 5.4. A dedicated server featuring an Intel Core i7-8700 CPU, 64 GB main memory and 12 TB secondary storage was used to host uploaded data sets and conduct necessary pre-processing operations. The applied processing pipeline was rather simplistic, combining just three pipeline nodes (Figure 6.7): An *importer* and an *exporter*, connected via a *kd-tree generator task*. To speed up performance, each kd-tree generator uses a main memory cache. In the context of this case study, the maximum cache size was set to 16 GB, thus, the number of kd-trees that could be generated in parallel was limited to four in the worst case (Figure 6.9). However, even for the largest uploaded data sets pre-processing times stayed below 60 minutes. Furthermore, they were added only gradually which further minimized the delay noticeable by the users.

6.2.2 Second Case Study

For the *second case study*, emphasis was put onto the processing engine's performance and scalability in more computation intense scenarios. In particular, the following scenarios were considered:

- **Deriving Tree Cadasters.** Tree cadasters consolidate detailed information about biomass in an area and are essential for a growing number of applications in

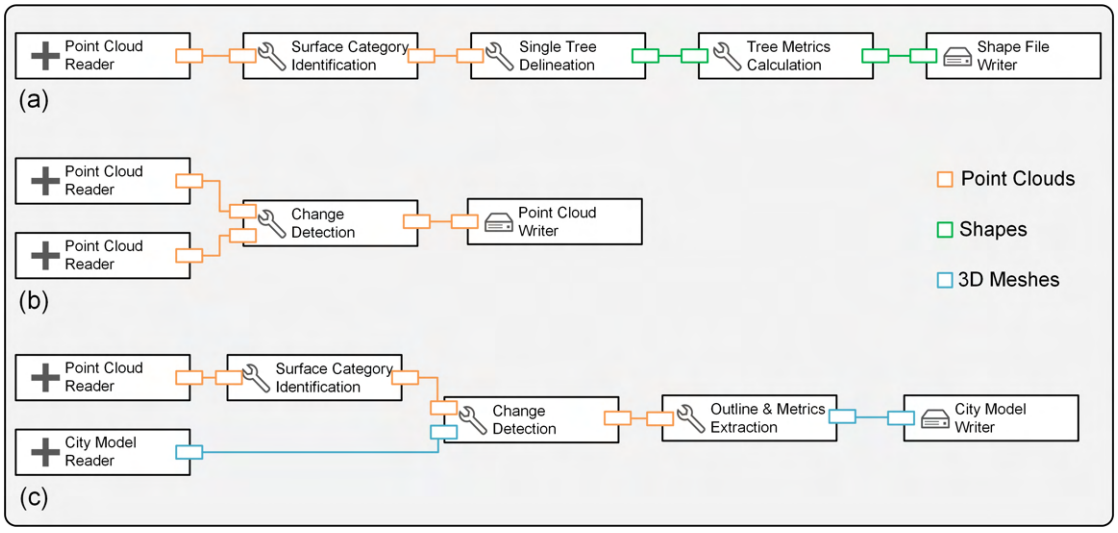


Figure 6.8: *Processing pipelines as they have been used for the second case study.*

urban planning, landscape architecture, and forest management. Point cloud analytics allows for the automatic, area-wide derivation and continuation of such tree cadasters and corresponding metrics, e.g., height and crown diameter [109]. As depicted in Figure 6.8(a), a corresponding analysis comprises three major tasks: Identification of points representing vegetation, delineation of individual trees within those vegetation points, and calculation of per-tree metrics. To identify vegetation points, an iterative, segment-based surface category extraction [139] was applied that distinguishes between ground, building, and vegetation by analyzing for each point the topology of its proximity. To efficiently delineate individual trees, a point-based approach by [91] was adapted, requiring only a single iteration over the point cloud. For fast nearest-neighbor queries a GPU-based implementation was used.

- **Monitoring Infrastructure Networks.** Infrastructure networks (e.g., roads, canalizations, or power lines) are constantly subjected to environmental loads (e.g., wind, temperature changes), causing them to deteriorate over time. Regularly capturing such infrastructure networks provides efficient means for their automatic, accurate, and predictive maintenance. The corresponding analysis (Figure 6.8(b)) is commonly referred to as a *change detection* [83, 57]: For each input point, the distance to a reference geometry (e.g., another point cloud or 3D model) is estimated as a metric for the degree of change within the covered area and stored as a per-point attribute. This approach allows to identify changes efficiently and establish update and maintaining workflows.

- **Continuing 3D City Models.** Many applications in urban planning and development or building information modeling require official, state-issued cadaster data or even full-fledged 3D city models. Keeping those models up to date constitutes

Table 6.1: *Second case study: Average data throughput of the processing engine.*

Processing Operation	Average Data Throughput
Surface Category Identification	0.44B pts/hour
Tree Delineation	0.53B pts/hour
Change Detection	7.68B pts/hour
Building Outline Extraction	8.27B pts/hour

a major challenge for municipalities that can be facilitated by using point cloud analytics. As an example, a change detection as described above can be combined with per-point surface category information to automatically identify all building points with a certain distance to a given 3D city model (Figure 6.8(c)), indicating newly constructed, re-moved, or otherwise modified buildings. The resulting points can be segmented into subsets of neighboring points, each representing a separate building. Based on these subsets, 2D building outlines and additional metrics (e.g., average distance, projected roof area) can be derived that facilitate the assessment of necessary —typically manual— modifications to the 3D city model.

For each scenario an individual pipeline plan (Section 2.5) was defined in the form of a JSON file to configure the processing pipeline's behavior accordingly. Users were then able to interactively inspect the processing results by switching between different rendering styles as described in Chapter 5.3. Test data sets for those scenarios comprised airborne, terrestrial, and mobile data sets covering different parts of three metropolitan regions with the largest point cloud featuring 100 points/m^2 for an area of 800 km^2. All tests and measurements were performed by a small network containing a total of six processing nodes, each featuring an Intel Core i7 CPU with 3.40 GHz, 32 GB main memory, and a NVIDIA GeForce GTX 1070 with 8 GB device memory and 1920 CUDA cores. In Table 6.1, the data throughput for the most relevant processing operations is specified. Data throughput for the integration and rendering of the data is defined by the overall network and memory bandwidth and was at around 80 MB/second.

6.2.3 Third Case Study

Following up on these measurements, a *third case study* further evaluated the processing pipeline's ability to scale with increased processing and memory capacities. Hosted on an Oracle Server —featuring an Intel Xeon Gold 5120M CPU, 192 GB main memory, 8 TB secondary storage, and two NVIDIA Tesla P100 with 16 GB device memory— a change detection as well as a surface category extraction were conducted for all uploaded data sets using the processing pipeline depicted in Figure 6.10. Test data for that case study consisted of two different airborne scans of a rural area of 270 km^2 featuring four points/m^2 (1.012 billion points in total) and nine points/m^2 (2.474 billion points in total), respectively. The computation time of both processing operations was notably

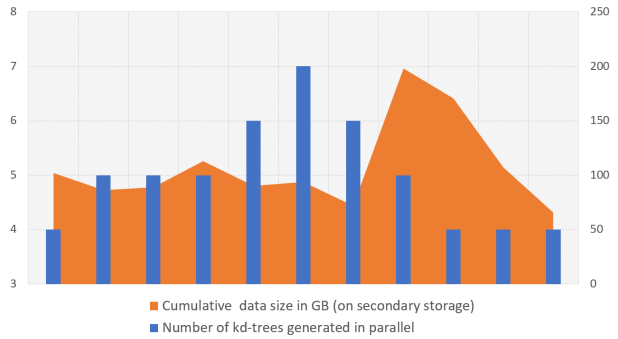


Figure 6.9: *Cache size was limited to 16 GB per kd-tree generator during the first case study. Hence, the number of kD-trees that could be generated in parallel (blue) varied depending on the overall size of the corresponding raw data (orange).*

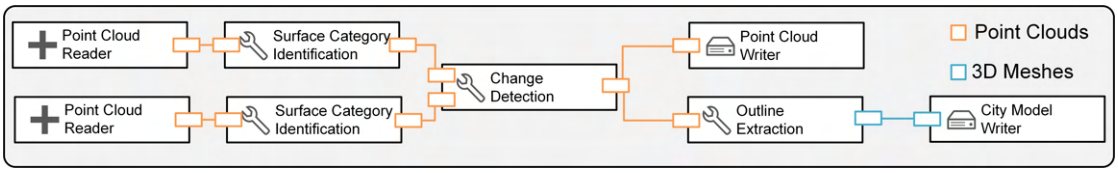
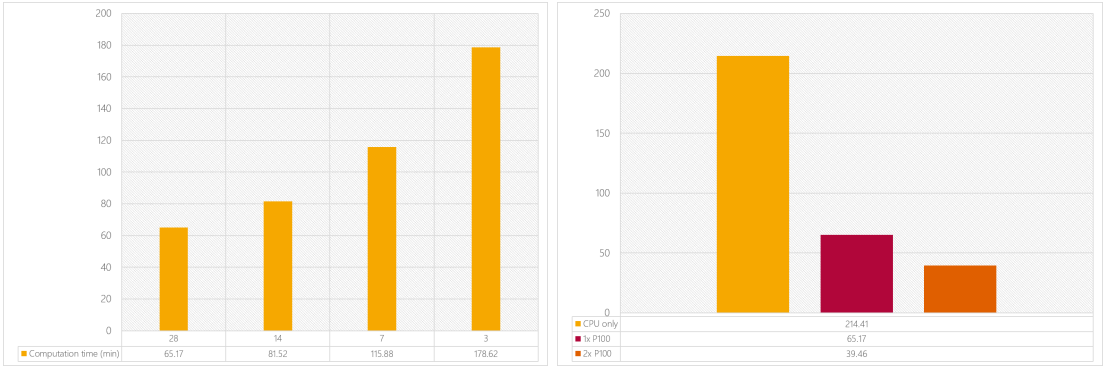


Figure 6.10: *Processing pipeline used for the third case study combining a change detection and a surface category identification.*



(a) *Computation time in minutes of the change detection for different numbers of CPU cores used (for one GPU).*

(b) *Computation time in minutes of the change detection for different numbers of GPUs used (for 28 CPU cores).*

Figure 6.11: *Third case study: Computation time in minutes of the change detection for different numbers of CPU and GPU cores.*

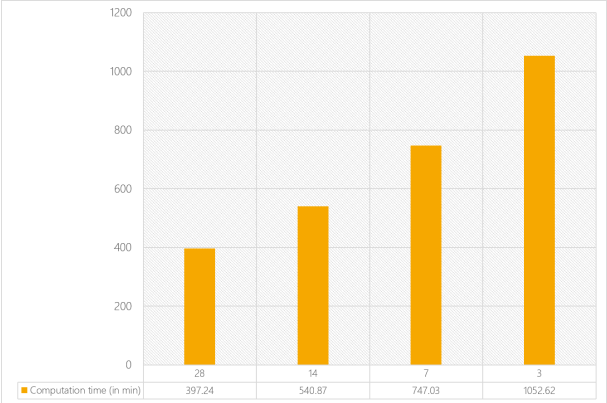


Figure 6.12: *Third case study: Computation time in minutes of the surface category identification for different numbers of CPU cores used (for one GPU).*

reduced by increasing the number of CPU threads used in parallel (Figure 6.11(a) and Figure 6.12) which underlines the scalability of the presented system. The performance of the change detection could be improved further by increasing the number of GPUs used during the computation (Figure 6.11(b)), whereas that had only a neglectable effect on the —in this case— mostly CPU based surface category extraction.

Chapter 7

Conclusions and Future Research

The rendering and interaction techniques presented throughout this thesis provide efficient means to design and implement visualization tools for enriched point clouds and 4D point clouds; they can be easily adapted for different use cases, target users and hardware platforms. In particular, they provide flexibility with respect to the following aspects:

- **Data Characteristic.** Depending on the applied acquisition system as well as subsequently conducted processing and analysis steps, captured point clouds can differ drastically in terms of covered surface area, local point density and number of data layers. As a consequence, the overall size of a visualized data set may lie anywhere between a few megabytes and hundreds of terabytes. Furthermore, additional data layers often necessitate applying unique color schemes and rendering styles to enable visual filtering and highlighting.
- **Hardware Limitations.** The maximum number of points that can be rendered simultaneously in real-time depends on the corresponding device's specific hardware setup, most notably available main and GPU memory as well as computation speed of CPU and GPU. Other factors to consider are memory and —in the case of web-based applications— network bandwidth, both influencing the speed and frequency at which visualized data and potential caches can be updated.
- **Targeted Frame Rate and Visual Quality.** The minimal frame rate required for an application to be considered usable varies based on the intended area of use: While traditional desktop applications are considered fully interactive at around 30 fps, VR applications require at least 90 fps to avoid nausea. Similarly, requirements regarding rendering style and visual quality (e.g., non-photorealistic vs. photorealistic) are often derived from the underlying use case.
- **Number of Simultaneous Users.** Especially large-scale acquisition projects focusing on entire cities or countries may involve several stakeholders of vastly different backgrounds that all need to interact with the captured data, may it be in the form of read-only operations (e.g., a simple visualization or the export of data subsets) or in a way that modifies the point cloud (e.g., via analysis and processing operations) and related visualization elements (e.g., by adding annotations or measurements). Visualization tools need to take this into account and ensure that the results of modifying operations are shared among all clients.

By combining the presented techniques, numerous users may collaboratively inspect enriched point clouds on heterogeneous hardware setups with differing computing capabilities, ranging from low-end mobile devices over high-end desktop computers to emerging VR devices. All techniques can be combined and configured at runtime, allowing to design task and domain-specific inspection tools, and can be seamlessly integrated with existing workflows and external systems. This is evidenced by a number of case studies and pilot user studies, showcasing the applicability of these techniques in the context of real-world scenarios.

In future work, some of the techniques discussed in this thesis may and should be further improved upon: First, the presented rendering techniques still require rather potent hardware (i.e., high-end dedicated or integrated GPUs) to provide interactive frame rates and, thus, exclude many hardware setups. To some degree, this can be seen as a result of all rendering techniques being based on the standard hardware-accelerated rendering pipeline implemented in modern GPUs, that has primarily been designed and optimized for mesh-based geometry rather than points – even though native point primitives are provided by underlying rendering systems (e.g., OpenGL's GL_POINTS). As a remedy, several authors [161, 73] have started to look into rendering techniques for point clouds that replace the standard hardware-accelerated rendering pipeline with custom compute shaders. Since such GPGPU-based rendering pipelines for 3D point clouds have the potential to *"outperform the hardware pipeline by up to an order of magnitude"* [161] while simultaneously improving the visual quality, it seems promising to adapt the rendering techniques presented in this thesis accordingly. Second, collaborative interaction has been primarily discussed from a *technical* point of view in this thesis by focusing on the implementation of interaction techniques allowing to inspect, measure and annotate as well as methods to synchronize corresponding interaction results across multiple devices. However, rather minimal focus has been put onto the *usability* of such interaction techniques. Therefore, the pilot user study presented in Chapter 4 should be followed up by additional full-fledged user studies that (a) take into account more diverse user groups, (b) evaluate the ability of the presented interaction techniques to facilitate interactions between multiple users and (c) focus on non-immersive applications and their drastically different requirements regarding usability.

Furthermore, the ideal of implementing visualization tools providing flexibility can be further evaluated with respect to the following aspects that were not further addressed in this thesis:

- **Data Stability.** Unlike the data sets used for evaluation purposes throughout this thesis, point clouds are not necessarily static but may be frequently updated, e.g., based on user interaction, processing results or regularly conducted re-acquisitions of the corresponding site. In particular, modifications may entail changes to the overall structure of a point cloud (i.e., by adding, moving, or removing entire points). This introduces additional challenges, as the spatial data structures and LoD representations utilized in previous chapters are generally optimized for fast data look-up at the cost of a significantly more complex construction and update

process. While spatial data structures and rendering techniques for dynamically modified point cloud have been introduced for desktop applications [157, 181], their applicability to web-based or VR applications remains to be evaluated.

- **Data Completeness.** The very nature of the capturing process often leads to varying point densities within a given data set: Depending on the positioning of the scanning device in relation to occluding structures as well as the reflection properties of captured surfaces, some parts of a site may be represented by considerably less points than others or even none at all. To some extent, such holes may be visually filled by assigning each point an ideal size specific to the point density in its proximity. However, calculating these ideal point sizes either requires additional preprocessing steps or slows down the rendering performance by adding complexity. An alternative that should be evaluated in future work is the use of deep learning rendering approaches [149, 16] that promise to efficiently *"generate high-resolution photo realistic point renderings from low-resolution point clouds"* [36].

An emerging technology that promises to expedite the digital transformation even further is known as *augmented reality (AR)*. While within this thesis, the concept of immersiveness was primarily discussed as an either-or property, differentiating between fully immersive visualizations based on VR technology and traditional non-immersive visualizations, AR technology has recently gained popularity as a bridge between both extremes, as it merely "blends computer-generated information on the user's real environment, while VR uses computer-generated information to provide a full sense of immersion" [117]. While AR applications for point clouds share certain requirements with web-based applications (e.g., having to scale for low-end mobile devices) as well as VR applications (e.g., being particularly vulnerable to visual artifacts that may break immersion), they also introduce several unique requirements that necessitate further research, in particular the precise localization of the digital model within the real-world environment.

Acknowledgements

This thesis is the result of my research at the Department of Computer Graphics Systems at the *Hasso Plattner Institute* (HPI). I am very grateful to my adviser Prof. Dr. Jürgen Döllner for granting me this opportunity.

As a research assistant at the HPI, in particular during my time as a member of the *HPI Research School for "Service-Oriented Systems Engineering"*, I had the opportunity to get in regular contact with a number of partners from industry and academia producing or operating on point clouds in some capacity. This allowed me to develop a deep understanding of the state-of-the-art in point cloud management, processing, and visualization as well as the most pressing needs encountered by professional users of corresponding applications, tools, and systems. In particular, it is a great pleasure to thank Dr. Rico Richter and Dr. Johannes Wolf for our joint research over the years that resulted in several publications. Furthermore, I'd like to thank Prof. James Gain, Prof. Heinz Rüther, Stephen Wessels, and Roshan Bhurtha from the *University of Cape Town* for their hospitality and in-depth discussion of the specific challenges of the *Zamani project*, a research group aiming to digitally preserve endangered cultural and natural heritage across Africa, the Middle East, and Southeast Asia via in-situ and remote sensing.

I owe sincere and earnest thankfulness to all the anonymous reviewers who supported me during writing this thesis by proofreading and by providing ideas for further improvement. This work has been partially funded by the HPI, the HPI Research School for "Service-Oriented Systems Engineering", and the German Federal Ministry of Education and Research (BMBF) as part of the "PunctumTube" research project.

List of Publications

The work presented in this manuscript appeared previously in the following publications:

Journal Papers and Book Chapters

[1] Sören Discher, Rico Richter, and Jürgen Döllner. "Concepts and Techniques for Web-Based Visualization and Processing of Massive 3D Point Clouds with Semantics". In: *Graphical Models* 104 (2019), p. 101036.

[2] Sören Discher, Rico Richter, Matthias Trapp, and Jürgen Döllner. "Service-Oriented Processing and Analysis of Massive Point Clouds in Geoinformation Management". In: *Service-Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management*. Springer International Publishing, 2019, pp. 43–61.

Conference Papers

[3] Sören Discher, Leon Masopust, Sebastian Schulz, Rico Richter, and Jürgen Döllner. "A Point-Based and Image-Based Multi-Pass Rendering Technique for Visualizing Massive 3D Point Clouds in VR Environments". In: *Proceedings of the 26. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. 2018.

[4] Sören Discher, Rico Richter, and Jürgen Döllner. "A Scalable WebGL-based Approach for Visualizing Massive 3D Point Clouds using Semantics-Dependent Rendering Techniques". In: *Proceedings of the 23rd International Conference on Web3D Technology*. 2018, pp. 1–9.

[5] Sören Discher, Rico Richter, and Jürgen Döllner. "Interactive and View-Dependent See-Through Lenses for Massive 3D Point Clouds". In: *Advances in 3D Geoinformation*. Springer International Publishing, 2016, pp. 49–62.

[6] Sören Discher, Rico Richter, and Jürgen Döllner. "Konzepte für eine Service-basierte Systemarchitektur zur Integration, Prozessierung und Analyse von massiven 3D-Punktwolken". In: *Tagungsband der 34. Wissenschaftlich-Technischen Jahrestagung der DGPF e.V.* 2019, pp. 1–10.

[7] Rico Richter, Sören Discher, and Jürgen Döllner. “Out-of-Core Visualization of Classified 3D Point Clouds”. In: *3D Geoinformation Science: The Selected Papers of the 3D GeoInfo*. Springer International Publishing, 2015, pp. 227–242.

[8] Felix Thiel, Sören Discher, Rico Richter, and Jürge Döllner. “Interaction and Locomotion Techniques for the Exploration of Massive 3D Point Clouds in VR Environments”. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 4*. 2018, pp. 697–701.

[9] Johannes Wolf, Sören Discher, and Jürgen Döllner. “Techniken zur kombinierten Darstellung von 2D-Bodenradar und 3D-Punktwolken zur Analyse des Straßenraums”. In: *Tagungsband der 39. Wissenschaftlich-Technischen Jahrestagung der DGPF e.V.* 2019, pp. 154–166.

[10] Johannes Wolf, Sören Discher, Leon Masopust, Sebastian Schulz, Rico Richter, and Jürgen Döllner. “Combined Visual Exploration of 2D Ground Radar and 3D Point Cloud Data for Road Environments”. In: *The International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 6210*. 2018, pp. 231–236.

[11] Johannes Wolf, Tobias Pietz, Rico Richter, Sören Discher, and Jürgen Döllner. “Image-Based Road Marking Classification and Vector Data Derivation from Mobile Mapping 3D Point Clouds”. In: *Proceedings of the 16th International Conference on Computer Graphics Theory and Applications*. 2021, pp. 227–234.

References

[12] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. AK Peters/crc Press, 2019.

[13] Varol Akman, W Randolph Franklin, Mohan Kankanhalli, and Chandrasekhar Narayanaswami. "Geometric Computing and Uniform Grid Technique". In: *Computer-Aided Design* 21.7 (1989), pp. 410–420.

[14] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. "Point Set Surfaces". In: *Proceedings of the IEEE Conference on Visualization*. IEEE. 2001, pp. 21–29.

[15] Fatemeh Alidoost and Hossein Arefi. "Comparison of UAS-Based Photogrammetry Software for 3D Point Cloud Generation: A Survey over a Historical Site". In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 4 (2017).

[16] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. "Neural Point-Based Graphics". In: *Proceedings of 16th European Conference on Computer Vision*. 2020, pp. 696–712.

[17] Thomas Alsop. *Virtual Reality (VR) - Statistics & Facts*. 2021. URL: `https://www.statista.com/topics/2532/virtual-reality-vr/#dossierKeyfigures` (visited on 12/02/2022).

[18] Carlos Andújar, Pere-Pau Vázquez, and Marta Fairén. "Way-Finder: Guided Tours through Complex Walkthrough Models". In: *Computer Graphics Forum*. Vol. 23. 3. 2004, pp. 499–508.

[19] Christoph Anthes, Rubén Jesús Garcıa-Hernández, Markus Wiedemann, and Dieter Kranzlmüller. "State of the Art of Virtual Reality Technology". In: *Proceedings of the IEEE Aerospace Conference*. IEEE. 2016, pp. 1–19.

[20] Salman Azhar. "Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry". In: *Leadership and Management in Engineering* 11.3 (2011), pp. 241–252.

[21] Esther Baumann, Fabrizio R Giorgetta, J-D Deschênes, William C Swann, Ian Coddington, and Nathan R Newbury. "Comb-Calibrated Laser Ranging for Three-Dimensional Surface Profiling with Micrometer-Level Precision at a Distance". In: *Optics express* 22.21 (2014), pp. 24914–24928.

[22] Jon Louis Bentley. “Multidimensional Binary Search Trees Used for Associative Searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[23] Jean-Angelo Beraldin, Michel Picard, Adriana Bandiera, Virginia Valzano, and Fabio Negro. “Best Practices for the 3D Documentation of the Grotta dei Cervi of Porto Badisco, Italy”. In: *Three-Dimensional Imaging, Interaction, and Measurement*. Vol. 7864. 2011, pp. 177–191.

[24] Leif P Berg and Judy M Vance. “Industry Use of Virtual Reality in Product Design and Manufacturing: A Survey”. In: *Virtual Reality* 21.1 (2017), pp. 1–17.

[25] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. “A survey of Surface Reconstruction from Point Clouds”. In: *Computer Graphics Forum*. Vol. 36. 1. 2017, pp. 301–329.

[26] Filip Biljecki, Ken Arroyo Ohori, Hugo Ledoux, Ravi Peters, and Jantien Stoter. “Population Estimation Using a 3D City Model: A Multi-Scale Country-Wide Study in the Netherlands”. In: *PloS one* 11.6 (2016), e0156808.

[27] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. “Applications of 3D City Models: State of the Art Review”. In: *ISPRS International Journal of Geo-Information* 4 (2015), pp. 2842–2889.

[28] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. “High-Quality Surface Splatting on Today’s GPUs”. In: *Eurographics Symposium on Point-Based Graphics*. 2005, pp. 17–24.

[29] Christian Boucheny. “Interactive Scientific Visualization of Large Datasets: Towards a Perceptive-Based Approach”. PhD thesis. Université Joseph Fourier, Grenoble, 2009.

[30] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. “Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks”. In: 2017, pp. 1–8.

[31] Mesude Bayrakci Boz, Kirby Calvert, and Jeffrey RS Brownson. “An Automated Model for Rooftop PV Systems Assessment in ArcGIS using LIDAR”. In: *Aims Energy* 3.3 (2015), pp. 401–420.

[32] E. Bozgeyikli, A. Raij, S. Katkoori, and R. Dubey. “Point & Teleport Locomotion Technique for Virtual Reality”. In: *Proceedings of the annual Symposium on Computer-Human Interaction in Play*. 2016, pp. 205–216.

[33] Alex Bradley, Haijiang Li, Robert Lark, and Simon Dunn. “BIM for Infrastructure: An Overall Review and Constructor Perspective”. In: *Automation in Construction* 71 (2016), pp. 139–152.

[34] John Brosz, Sheelagh Carpendale, and Miguel A Nacenta. “The Undistort Lens”. In: *Computer Graphics Forum*. Vol. 30. 3. 2011, pp. 881–890.

[35] John Brosz, Faramarz F Samavati, M Sheelagh T Carpendale, and Mario Costa Sousa. “Single Camera Flexible Projection”. In: *Proceedings of the 5th international Symposium on Non-Photorealistic Animation and Rendering*. 2007, pp. 33–42.

[36] Giang Bui, Truc Le, Brittany Morago, and Ye Duan. “Point-Based Rendering Enhancement via Deep Learning”. In: *The Visual Computer* 34.6 (2018), pp. 829–841.

[37] Nicholas Burtnyk, Azam Khan, George Fitzmaurice, Ravin Balakrishnan, and Gordon Kurtenbach. “Stylecam: Interactive Stylized 3D Navigation using Integrated Spatial & Temporal Controls”. In: *Proceedings of the 15th annual ACM Symposium on User Interface Software and Technology*. 2002, pp. 101–110.

[38] Howard Butler, David C Finnegan, Peter J Gadomski, and Uday K Verma. “plas. io: Open Source, Browser-Based WebGL Point Cloud Visualization”. In: *AGU Fall Meeting Abstracts* 2014 (2014), IN23D–3749.

[39] Matthew Carlberg, Peiran Gao, George Chen, and Avideh Zakhor. “Classifying Urban Landscape in Aerial Lidar Using 3D Shape Analysis”. In: *16th IEEE International Conference on Image Processing*. 2009, pp. 1701–1704.

[40] Dong Chen, Ruisheng Wang, and Jiju Peethambaran. “Topologically Aware Building Rooftop Reconstruction from Airborne Laser Scanning Point Clouds”. In: *IEEE Transactions on Geoscience and Remote Sensing* 55.12 (2017), pp. 7032–7052.

[41] Haiwei Chen, Samantha Chen, and Evan Suma Rosenberg. “Redirected Walking in Irregularly Shaped Physical Environments with Dynamic Obstacles”. In: *Proceedings of the IEEE Conference on Virtual Reality and 3D User Interfaces*. 2018. Forthcoming.

[42] K Choromański, J Łobodecki, K Puchała, and W Ostrowski. “Development of Virtual Reality Application for Cultural Heritage Visualization From Multi-Source 3D Data”. In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).

[43] Martin Christen and Stephan Nebiker. “Visualisation of Complex 3D City Models on Mobile Webbrowsers using Cloud-Based Image Provisioning”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2015).

[44] Rémi Cura, Julien Perret, and Nicolas Paparoditis. “A Scalable and Multi-Purpose Point Cloud Server (PCS) for Easier and Faster Point Cloud Data Management and Processing”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 127 (2017), pp. 39–56.

[45] Paolo Dabove, Nives Grasso, and Marco Piras. “Smartphone-Based Photogrammetry for the 3D Modeling of a Geomorphological Structure”. In: *Applied Sciences* 9.18 (2019), p. 3884.

[46] James L Davis and A Peter Annan. “Ground-Penetrating Radar for High-Resolution Mapping of Soil and Rock Stratigraphy”. In: *Geophysical prospecting* 37.5 (1989), pp. 531–551.

[47] M De La Calle, D Gómez-Deck, O Koehler, and F Pulido. "Point Cloud Visualization in an Open Source 3D glob3". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38.5/W16 (2011).

[48] David Deibe, Margarita Amor, and Ramón Doallo. "Big Data Storage Technologies: A Case Study for Web-Based LiDAR Visualization". In: *Proceedings of the IEEE International Conference on Big Data*. IEEE. 2018, pp. 3831–3840.

[49] David Deibe, Margarita Amor, and Ramón Doallo. "Supporting Multi-Resolution Out-of-Core Rendering of Massive LiDAR Point Clouds through Non-Redundant Data Structures". In: *International Journal of Geographical Information Science* 33.3 (2019), pp. 593–617.

[50] David Deibe, Margarita Amor, Ramón Doallo, David Miranda, and Miguel Cordero. "GVLiDAR: An Interactive Web-Based Visualization Framework to Support Geospatial Measures on LiDAR Data". In: *International Journal of Remote Sensing* 38.3 (2017), pp. 827–849.

[51] Emanuel Demetrescu, Enzo d'Annibale, Daniele Ferdani, and Bruno Fanini. "Digital Replica of Cultural Landscapes: An Experimental Reality-Based Workflow to Create Realistic, Interactive Open World Experiences". In: *Journal of Cultural Heritage* 41 (2020), pp. 125–141.

[52] Petar Dobrev, Paul Rosenthal, and Lars Linsen. "An Image-Space Approach to Interactive Point Cloud Rendering Including Shadows and Transparency". In: *Computer Graphics and Geometry* 12.3 (2010), pp. 2–25.

[53] Juergen Dold and Jessica Groopman. "The Future of Geospatial Intelligence". In: *Geo-Spatial Information Science* 20.2 (2017), pp. 151–162.

[54] Jürgen Döllner. "Geospatial Artificial Intelligence: Potentials of Machine Learning for 3D Point Clouds and Geospatial Digital Twins". In: *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88.1 (2020), pp. 15–24.

[55] Jürgen Döllner, Benjamin Hagedorn, and Jan Klimke. "Server-Based Rendering of Large 3D Scenes for Mobile Devices using G-Buffer Cube Maps". In: *Proceedings of the 17th International Conference on 3D Web Technology*. 2012, pp. 97–100.

[56] Pinliang Dong and Qi Chen. *LiDAR Remote Sensing and Applications*. CRC Press, 2017.

[57] Jan UH Eitel, Bernhard Höfle, Lee A Vierling, Antonio Abellán, Gregory P Asner, Jeffrey S Deems, Craig L Glennie, Philip C Joerg, Adam L LeWinter, Troy S Magney, *et al.* "Beyond 3-D: The New Spectrum of Lidar Applications for Earth and Ecological Sciences". In: *Remote Sensing of Environment* 186 (2016), pp. 372–392.

[58] Niklas Elmqvist and Philippas Tsigas. "A Taxonomy of 3D Occlusion Management for Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 14.5 (2008), pp. 1095–1109.

[59] Jan Elseberg, Dorit Borrmann, and Andreas Nüchter. “One Billion Points in the Cloud – An Octree for Efficient Processing of 3D Laser Scans”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 76 (2013), pp. 76–88.

[60] Karim Farghaly, Fonbeyin Henry Abanda, Christos Vidalakis, and Graham Wood. “Taxonomy for BIM and Asset Management Semantic Interoperability”. In: *Journal of Management in Engineering* 34.4 (2018), pp. 1–13.

[61] David N Ford and Charles M Wolf. “Smart Cities with Digital Twin Systems for Disaster Management”. In: *Journal of Management in Engineering* 36.4 (2020), p. 04020027.

[62] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. “Digital Twin: Enabling Technologies, Challenges and Open Research”. In: *IEEE Access* 8 (2020), pp. 108952–108971.

[63] Xiang Gao, Hainan Cui, Lingjie Zhu, Tianxin Shi, and Shuhan Shen. “Multi-Source Data-Based 3D Digital Preservation of Largescale Ancient Chinese Architecture: A Case Report”. In: *Virtual Reality & Intelligent Hardware* 1.5 (2019), pp. 525–541.

[64] Zhenzhen Gao, Luciano Nocera, and Ulrich Neumann. “Visually-Complete Aerial LiDAR Point Cloud Rendering”. In: *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. 2012, pp. 289–298.

[65] Ali Ghaffarianhoseini, John Tookey, Amirhosein Ghaffarianhoseini, Nicola Naismith, Salman Azhar, Olia Efimova, and Kaamran Raahemifar. “Building Information Modelling (BIM) Uptake: Clear Benefits, Understanding its Implementation, Risks and Challenges”. In: *Renewable and Sustainable Energy Reviews* 75 (2017), pp. 1046–1053.

[66] Enrico Gobbetti and Fabio Marton. “Layered Point Clouds: A Simple and Efficient Multiresolution Structure for Distributing and Rendering Gigantic Point-Sampled Models”. In: *Computers & Graphics* 28.6 (2004), pp. 815–826.

[67] Michael Goesele, Jens Ackermann, Simon Fuhrmann, Carsten Haubold, Ronny Klowsky, Drew Steedly, and Richard Szeliski. “Ambient point clouds for view interpolation”. In: *ACM Transactions on Graphics* 29.4 (2010), 95:1–95:6.

[68] Leonardo Gomes, Olga Regina Pereira Bellon, and Luciano Silva. “3D Reconstruction Methods for Digital Preservation of Cultural Heritage: A Survey”. In: *Pattern Recognition Letters* 50 (2014), pp. 3–14.

[69] Prashant Goswami, Fatih Erol, Rahul Mukhi, Renato Pajarola, and Enrico Gobbetti. “An Efficient Multi-Resolution Framework for High Quality Interactive Rendering of Massive Point Clouds using Multi-Way kd-Trees”. In: *The Visual Computer* 29.1 (2013), pp. 69–83.

[70] Chris Green. “Improved Alpha-tested Magnification for Vector Textures and Special Effects”. In: *Proceedings of ACM SIGGRAPH 2007 Courses*. 2007, pp. 9–18.

[71] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Elsevier, 2011.

[72] Alberto Guarnieri, Nicola Milan, and Antonio Vettore. "Monitoring of Complex Structure for Structural Control Using Terrestrial Laser Scanning (TLS) and Photogrammetry". In: *International Journal of Architectural Heritage* 7.1 (2013), pp. 54–67.

[73] Christian Günther, Thomas Kanzok, Lars Linsen, and Paul Rosenthal. "A GPGPU-Based Pipeline for Accelerated Rendering of Point Clouds". In: 21 (2013), pp. 153–161.

[74] Tanishq Gupta and Holden Li. "Indoor Mapping for Smart Cities – An Affordable Approach: Using Kinect Sensor and ZED Stereo Camera". In: *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation*. IEEE. 2017, pp. 1–8.

[75] Ralf Gutbell, Lars Pandikow, Volker Coors, and Yasmina Kammeyer. "A Framework for Server Side Rendering using OGC's 3D Portrayal Service". In: *Proceedings of the 21st International Conference on Web3D Technology*. 2016, pp. 137–146.

[76] Benjamin Hagedorn, Simon Thum, Thorsten Reitz, Volker Coors, and Ralf Gutbell. *OGC 3D Portrayal Service 1.0*. OGC Implementation Standard 1.0. Open Geospatial Consortium, Sept. 2017.

[77] Mahmudul Hasan, Faramarz F Samavati, and Christian Jacob. "Multilevel Focus+Context Visualization using Balanced Multiresolution". In: *Proceedings of the International Conference on Cyberworlds*. IEEE. 2014, pp. 145–152.

[78] John F Hughes, Andries Van Dam, Morgan McGuire, David F Sklar, James D Foley, Steven K Feiner, and Kurt Akeley. *Computer Graphics: Principles and Practice (3rd ed.)* Addison-Wesley Professional, 2013.

[79] Dryver R Huston, Noel V Pelczarski, Brian Esser, and Kenneth R Maser. "Damage Detection in Roadways with Ground Penetrating Radar". In: *Proceedings of the 8th International Conference on Ground Penetrating Radar*. Vol. 4084. 2000, pp. 91–94.

[80] Young Hoon Jo and Seonghyuk Hong. "Three-Dimensional Digital Documentation of Cultural Heritage Site Based on the Convergence of Terrestrial Laser Scanning and Unmanned Aerial Vehicle Photogrammetry". In: *ISPRS International Journal of Geo-Information* 8.2 (2019), pp. 53–66.

[81] Andreas Jochem, Bernhard Höfle, Volker Wichmann, Martin Rutzinger, and Alexander Zipf. "Area-Wide Roof Plane Segmentation in Airborne LiDAR Point Clouds". In: *Computers, Environment and Urban Systems* 36.1 (2012), pp. 54–64.

[82] Mikael Johansson. "Efficient Stereoscopic Rendering of Building Information Models (BIM)". In: *Journal of Computer Graphics Techniques* 5.3 (2016).

[83] Zhizhong Kang and Zhao Lu. “The Change Detection of Building Models using Epochs of Terrestrial Point Clouds”. In: *Proceedings of the International Workshop on Multi-Platform/Multi-Sensor Remote Sensing and Mapping*. IEEE. 2011, pp. 1–6.

[84] Thomas P Kersten, H-J Przybilla, Maren Lindstaedt, Felix Tschirschwitz, and Martin Misgaiski-Hass. “Comparative Geometrical Investigations of Hand-Held Scanning Systems”. In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 41 (2016).

[85] Hyeon-Joong Kim, A. Cengiz Öztireli, Markus Gross, and Soo-Mi Choi. “Adaptive Surface Splatting for Facial Rendering”. In: *Computer Animation and Virtual Worlds* 23.3-4 (2012), pp. 363–373.

[86] Michel Krämer and Ralf Gutbell. “A Case Study on 3D Geospatial Applications in the Web using State-of-the-Art WebGL Frameworks”. In: *Proceedings of the 20th International Conference on 3D Web Technology*. 2015, pp. 189–197.

[87] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. “Digital Twin in Manufacturing: A Categorical Literature Review and Classification”. In: *IFAC PapersOnLine* 51.11 (2018), pp. 1016–1022.

[88] Jeff de La Beaujardiere. “OpenGIS® Web Map Server Implementation Specification. Version 1.3.0”. In: (2006).

[89] Tobias Langner, Daniel Seifert, Bennet Fischer, Daniel Goehring, Tinosch Ganjineh, and Raúl Rojas. “Traffic Awareness Driver Assistance Based on Stereovision, Eye-Tracking, and Head-Up Display”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 3167–3173.

[90] Franz Leberl, Arnold Irschara, Thomas Pock, Philipp Meixner, Michael Gruber, Set Scholz, and Alexander Wiechert. “Point Clouds: Lidar versus 3D Vision”. In: *Photogrammetric Engineering & Remote Sensing* 76.10 (2010), pp. 1123–1134.

[91] Wenkai Li, Qinghua Guo, Marek K Jakubowski, and Maggi Kelly. “A New Method for Segmenting Individual Trees from the Lidar Point Cloud”. In: *Photogrammetric Engineering & Remote Sensing* 78.1 (2012), pp. 75–84.

[92] Ying Liu, Lin Zhang, Yuan Yang, Longfei Zhou, Lei Ren, Fei Wang, Rong Liu, Zhibo Pang, and M Jamal Deen. “A Novel Cloud-Based Framework for the Elderly Healthcare Services Using Digital Twin”. In: *IEEE Access* 7 (2019), pp. 49088–49101.

[93] Yuanxin Liu and Jack Snoeyink. “A Comparison of Five Implementations of 3D Delaunay Tessellation”. In: *Combinatorial and Computational Geometry* 52 (2005), pp. 439–458.

[94] Suresh K. Lodha, Darren M. Fitzpatrick, and David P. Helmbold. “Aerial Lidar Data Classification using AdaBoost”. In: *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM)*. 2007, pp. 435–442.

[95] Facundo José López, Pedro M Lerones, José Llamas, Jaime Gómez-Garcıa-Bermejo, and Eduardo Zalama. “A Review of Heritage Building Information Modeling (H-BIM)”. In: *Multimodal Technologies and Interaction* 2.21 (2018), pp. 1–29.

[96] Alexey Lukin. “Tips & Tricks: Fast Image Filtering Algorithms”. In: *Proceedings of GraphiCon*. 2007, pp. 186–189.

[97] Ryan Magargle, Lee Johnson, Padmesh Mandloi, Peyman Davoudabadi, Omkar Kesarkar, Sivasubramani Krishnaswamy, John Batteh, and Anand Pitchaikani. “A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System”. In: *Proceedings of the 12th International Modelica Conference*. 2017, pp. 35–46.

[98] Oscar Martinez-Rubi, Stefan Verhoeven, Maarten Van Meersbergen, Peter Van Oosterom, R GonÁalves, Theo Tijssen, *et al.* “Taming the Beast: Free and Open-Source Massive Point Cloud Web Visualization”. In: *Proceedings of the Capturing Reality Forum*. The Servey Association. 2015.

[99] Adam Marx. “Using Metaphor Effectively in User Interface Design”. In: *Proceedings of Conference on Human factors in Computing Systems*. 1994, pp. 379–380.

[100] Jane Matthews, Peter ED Love, Joshua Mewburn, Christopher Stobaus, and Chamila Ramanayaka. “Building Information Modelling in Construction: Insights from Collaboration and Change Management Perspectives”. In: *Production Planning & Control* 29.3 (2018), pp. 202–216.

[101] Dimitri J Mavriplis. “An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness”. In: *Journal of Computational Physics* 117.1 (1995), pp. 90–101.

[102] Donald Meagher. “Geometric Modeling Using Octree Encoding”. In: *Computer graphics and image processing* 19.2 (1982), pp. 129–147.

[103] Niloy J. Mitra and An Nguyen. “Estimating Surface Normals in Noisy Point Cloud Data”. In: *Proceedings of the 19th Annual Symposium on Computational Geometry*. 2003, pp. 322–328.

[104] Martin Mittring. “Finding Next Gen: Cryengine 2”. In: *ACM SIGGRAPH 2007 courses*. 2007, pp. 97–121.

[105] Matthias Mueller and Benjamin Pross. “OGC WPS 2.0 Interface Standard. Version 2.0”. In: (2015).

[106] Stephan Nebiker, Susanne Bleisch, and Martin Christen. “Rich Point Clouds in Virtual Globes – A New Paradigm in City Modeling?” In: *Computers, Environment and Urban Systems* 34.6 (2010), pp. 508–517.

[107] Marc Nienhaus and Jürgen Döllner. “Blueprint Rendering and Sketchy Drawings”. In: *GPU Gems* 2 (2005), pp. 235–252.

[108] Edwin Nissen, Tadashi Maruyama, J Ramon Arrowsmith, John R Elliott, Aravindhan K Krishnan, Michael E Oskin, and Srikanth Saripalli. "Coseismic Fault Zone Deformation Revealed with Differential Lidar: Examples from Japanese Mw 7 Intraplate Earthquakes". In: *Earth and Planetary Science Letters* 405 (2014), pp. 244–256.

[109] Christoph Oehlke, Rico Richter, and Jürgen Döllner. "Automatic Detection and Large-Scale Visualization of Trees for Digital Landscapes and City Models Based on 3D Point Clouds". In: *Proceedings of the 16th Conference on Digital Landscape Architecture*. 2015, pp. 151–160.

[110] M Olson, Ramsay Dyer, Hao Zhang, and Alla Sheffer. "Point Set Silhouettes via Local Reconstruction". In: *Computers & Graphics* 35.3 (2011), pp. 500–509.

[111] Hany Omar, Lamine Mahdjoubi, and Gamal Kheder. "Towards an Automated Photogrammetry-Based Approach for Monitoring and Controlling Construction Site Activities". In: *Computers in Industry* 98 (2018), pp. 172–182.

[112] Peter van Oosterom, Oscar Martinez-Rubi, Milena Ivanova, Mike Horhammer, Daniel Geringer, Siva Ravada, Theo Tijssen, Martin Kodde, and Romulo Gonçalves. "Massive Point Cloud Data Management: Design, Implementation and Execution of a Point Cloud Benchmark". In: *Computers & Graphics* 49 (2015), pp. 92–125.

[113] Francesco Osti, Raffaele de Amicis, Christopher A Sanchez, Azara Betony Tilt, Eric Prather, and Alfredo Liverani. "A VR Training System for Learning and Skills Development for Construction Workers". In: *Virtual Reality* 25.2 (2021), pp. 523–538.

[114] Steve Ostrowski, G Jozkow, Charles Toth, and Benjamin Vander Jagt. "Analysis of Point Cloud Generation from UAS Images". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2.1 (2014), p. 45.

[115] Johannes Otepka, Sajid Ghuffar, Christoph Waldhauser, Ronald Hochreiter, and Norbert Pfeifer. "Georeferenced Point Clouds: A Survey of Features and Point Cloud Management". In: *ISPRS International Journal of Geo-Information* 2.4 (2013), pp. 1038–1065.

[116] Alice Paladini, Abhijit Dhanda, Miquel Reina Ortiz, Adam Weigert, Eslam Nofal, A Min, M Gyi, S Su, Koen Van Balen, and Mario Santana Quintero. "Impact of Virtual Reality Experience on Accessibility of Cultural Heritage". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42.2/W11 (2019), pp. 929–936.

[117] Vista Equity Partners. *An Introduction to Immersive Technologies*. `https://www.vistaequitypartners.com/insights/an-introduction-to-immersive-technologies/`. Accessed: 2022-12-02. 2022.

[118] Sebastian Pasewaldt, Amir Semmo, Matthias Trapp, and Jürgen Döllner. "Multi-Perspective 3D Panoramas". In: *International Journal of Geographical Information Science* 28.10 (2014), pp. 2030–2051.

[119] Paola Passalacqua, Patrick Belmont, Dennis M Staley, Jeffrey D Simley, J Ramon Arrowsmith, Collin A Bode, Christopher Crosby, Stephen B DeLong, Nancy F Glenn, Sara A Kelly, *et al.* "Analyzing High Resolution Topography for Advancing the Understanding of Mass and Energy Transfer Through Landscapes: A Review". In: *Earth-Science Reviews* 148 (2015), pp. 174–193.

[120] George Pavlidis, Anestis Koutsoudis, Fotis Arnaoutoglou, Vassilios Tsioukas, and Christodoulos Chamzas. "Methods for 3D Digitization of Cultural Heritage". In: *Journal of Cultural Heritage* 8.1 (2007), pp. 93–98.

[121] Shmuel Peleg, Moshe Ben-Ezra, and Yael Pritch. "Omnistereo: Panoramic Stereo Imaging". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.3 (2001), pp. 279–290.

[122] Jorge Posada, Mikel Zorrilla, Ana Dominguez, Bruno Simoes, Peter Eisert, Didier Stricker, Jason Rambach, Jürgen Döllner, and Miguel Guevara. "Graphics and Media Technologies for Operators in Industry 4.0". In: *IEEE Computer Graphics and Applications* 38.5 (2018), pp. 119–132.

[123] Florent Poux, Roland Billen, Jean-Paul Kasprzyk, Pierre-Henri Lefebvre, and Pierre Hallot. "A Built Heritage Information System Based on Point Cloud Data: HIS-PC". In: *ISPRS International Journal of Geo-Information* 9.10 (2020), p. 588.

[124] Florent Poux, Romain Neuville, Pierre Hallot, Line Van Wersch, Andrea Luczfalvy Jancsó, and Roland Billen. "Digital Investigations of an Archaeological Smart Point Cloud: A Real Time Web-Based Platform to Manage the Visualisation of Semantical Queries". In: *Conservation of Cultural Heritage in the Digital Era* (2017), pp. 581–588.

[125] Florent Poux, Romain Neuville, Gilles-Antoine Nys, and Roland Billen. "3D Point Cloud Semantic Modelling: Integrated Framework for Indoor Spaces and Furniture". In: *Remote Sensing* 10.9 (2018), p. 1412.

[126] Federico Prandi, Federico Devigili, Marco Soave, Umberto Di Staso, and Raffaele De Amicis. "3D Web Visualization of Huge CityGML Models". In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 40 (2015).

[127] Reinhold Preiner, Stefan Jeschke, and Michael Wimmer. "Auto Splats: Dynamic Point Cloud Visualization on the GPU". In: *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization*. 2012, pp. 139–148.

[128] I Puente, H González-Jorge, J Martınez-Sánchez, and P Arias. "Review of Mobile Mapping and Surveying Technologies". In: *Measurement* 46.7 (2013), pp. 2127–2145.

[129] Imad L Al-Qadi and Samer Lahouar. “Measuring Layer Thicknesses with GPR – Theory to practice”. In: *Construction and Building Materials* 19.10 (2005), pp. 763–772.

[130] Sharif Razzaque. “Redirected Walking”. PhD thesis. 2005.

[131] Sharif Razzaque, Zachariah Kohn, and Mary C Whitton. “Redirected Walking”. In: *Proceedings of EUROGRAPHICS*. 2001, pp. 105–106.

[132] Sharif Razzaque, David Swapp, Mel Slater, Mary C Whitton, and Anthony Steed. “Redirected Walking in Place”. In: *Proceedings of the 8th Eurographics Workshop on Virtual Environments*. 2002, pp. 123–130.

[133] Tyler Read, Christopher A Sanchez, and Raffaele De Amicis. “Engagement and Time Perception in Virtual Reality”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 65. 1. SAGE Publications Sage CA: Los Angeles, CA. 2021, pp. 913–918.

[134] Danijel Rebolj, Zoran Pučko, Nenad Čuš Babič, Marko Bizjak, and Domen Mongus. “Point Cloud Quality Requirements for Scan-vs-BIM Based Automated Construction Progress Monitoring”. In: *Automation in Construction* 84 (2017), pp. 323–334.

[135] Fabio Remondino, Fabio Menna, Anestis Koutsoudis, Christos Chamzas, and Sabry El-Hakim. “Design and Implement a Reality-Based 3D Digitisation and Modelling Project”. In: *Proceedings of the Digital Heritage International Congress*. Vol. 1. IEEE. 2013, pp. 137–144.

[136] Fabio Remondino, Maria Grazia Spera, Erica Nocerino, Fabio Menna, and Francesco Nex. “State of the Art in High Density Image Matching”. In: *The photogrammetric record* 29.146 (2014), pp. 144–166.

[137] Fabio Remondino, Maria Grazia Spera, Erica Nocerino, Fabio Menna, Francesco Nex, and Sara Gonizzi-Barsanti. “Dense Image Matching: Comparisons and Analyses”. In: *Proceedings of the Digital Heritage International Congress*. Vol. 1. IEEE. 2013, pp. 47–54.

[138] Rico Richter. “Concepts and Techniques for Processing and Rendering of Massive 3D Point Clouds”. PhD thesis. Universität Potsdam, 2018.

[139] Rico Richter, Markus Behrens, and Jürgen Döllner. “Object Class Segmentation of Massive 3D Point Clouds of Urban Areas Using Point Cloud Topology”. In: *International Journal of Remote Sensing* 34.23 (2013), pp. 8408–8424.

[140] Rico Richter and Jürgen Döllner. “Concepts and Techniques for Integration, Analysis and Visualization of Massive 3D Point Clouds”. In: *Computers, Environment and Urban Systems* 45 (2014), pp. 114–124.

[141] Rico Richter and Jürgen Döllner. “Out-of-core Real-time Visualization of Massive 3D Point Clouds”. In: *Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. 2010, pp. 121–128.

[142] Rico Richter, Jan Eric Kyprianidis, and Jürgen Döllner. “Out-of-Core GPU-Based Change Detection in Massive 3D Point Clouds”. In: *Transactions in GIS* 17.5 (2013), pp. 724–741.

[143] Marcos Balsa Rodriguez, Enrico Gobbetti, Fabio Marton, Ruggero Pintus, Giovanni Pintore, and Alex Tinti. “Interactive Exploration of Gigantic Point Clouds on Mobile Devices”. In: *Proceedings of the 13th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*. 2012, pp. 57–64.

[144] Pablo Rodrıguez-Gonzálvez, Belen Jimenez Fernandez-Palacios, Ángel Luis Muñoz-Nieto, Pedro Arias-Sanchez, and Diego Gonzalez-Aguilera. “Mobile LiDAR System: New Possibilities for the Documentation and Dissemination of Large Cultural Heritage Sites”. In: *Remote Sensing* 9.3 (2017), p. 189.

[145] Timo Ropinski, Klaus Hinrichs, and Frank Steinicke. “A Solution for the Focus and Context Problem in Interactive Geovisualization Applications”. In: *Proceedings of the DMGIS Workshop on Dynamic and Multi-dimensional GIS*. 2005, pp. 144–149.

[146] Paul Rosenthal and Lars Linsen. “Image-Space Point Cloud Rendering”. In: *Proceedings of Computer Graphics International*. 2008, pp. 136–143.

[147] Jürgen Roßmann, Martin Hoppen, and Arno Bücken. “GML-Based Data Management and Semantic World Modelling for a 4D Forest Simulation and Information System”. In: *International Journal of 3-D Information Modeling* 3.3 (2014), pp. 50–67.

[148] Mathias Rothermel, Konrad Wenzel, Dieter Fritsch, and Norbert Haala. “SURE: Photogrammetric Surface Reconstruction from Imagery”. In: *Proceedings of the LC3D Workshop*. Vol. 8. 2. 2012.

[149] Darius Rückert, Linus Franke, and Marc Stamminger. “Adop: Approximate Differentiable One-Pixel Point Rendering”. In: *ACM Transactions on Graphics* 41.4 (2022), pp. 1–14.

[150] S Rusinkiewicz and M Levoy. “QSplat: A Multiresolution Point Rendering System for Large Meshes”. In: *Proceedings of ACM SIGGRAPH*. 2000, pp. 343–352.

[151] Heinz Rüther, Michael Chazan, Ralph Schroeder, Rudy Neeser, Christoph Held, Steven James Walker, Ari Matmon, and Liora Kolska Horwitz. “Laser Scanning for Conservation and Research of African Cultural Heritage Sites: The Case Study of Wonderwerk Cave, South Africa”. In: *Journal of Archaeological Science* 36.9 (2009), pp. 1847–1856.

[152] Takafumi Saito and Tokiichiro Takahashi. “Comprehensible Rendering of 3-D Shapes”. In: *SIGGRAPH Computer Graphics* 24.4 (1990), pp. 197–206.

[153] Hanan Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[154] Hanan Samet. “The Quadtree and Related Hierarchical Data Structures”. In: *ACM Computing Surveys* 16.2 (1984), pp. 187–260.

[155] Bhuvaneswari Sarupuri, Simon Hoermann, Mary C Whitton, and Robert W Lindeman. “Evaluating and Comparing Game-controller based Virtual Locomotion Techniques”. In: *Proceedings of the Eurographics Symposium on Virtual Environments*. 2017.

[156] Claus Scheiblauer. “Interactions with Gigantic Point Clouds”. PhD thesis. 2014.

[157] Claus Scheiblauer and Michael Wimmer. “Out-of-Core Selection and Editing of Huge Point Clouds”. In: *Computers & Graphics* 35.2 (2011), pp. 342–351.

[158] Alexander Schoedon, Matthias Trapp, Henning Hollburg, and Jürgen Döllner. “Interactive Web-Based Visualization for Accessibility Mapping of Transportation Networks”. In: *Proceedings of the Eurographics Conference on Visualization*. 2016, pp. 79–83.

[159] M. Schütz and M. Wimmer. “Rendering Large Point Clouds in Web Browsers”. In: *Proceedings of the 19th Central European Seminar on Computer Graphics* (2015), pp. 83–90.

[160] Markus Schütz. “Massive Time-Lapse Point Cloud Rendering in Virtual Reality”. In: *Presentation at ACM SIGGRAPH* (2016).

[161] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. “Rendering Point Clouds with Compute Shaders and Vertex Order Optimization”. In: *Computer Graphics Forum* 40.4 (2021), pp. 115–126.

[162] Markus Schütz and Michael Wimmer. “High-Quality Point-Based Rendering Using Fast Single-Pass Interpolation”. In: *Proceedings of the Digital Heritage International Congress*. Vol. 1. IEEE. 2015, pp. 369–372.

[163] Dominik Sibbing, Torsten Sattler, Bastian Leibe, and Leif Kobbelt. “SIFT-Realistic Rendering”. In: *Proceedings of the International Conference on 3D Vision* (2013), pp. 56–63.

[164] Stephan Sigg, Raphael Fuchs, Robert Carnecky, and Ronald Peikert. “Intelligent Cutaway Illustrations”. In: *Proceedings of the IEEE Pacific Visualization Symposium*. IEEE. 2012, pp. 185–192.

[165] Mel Slater, Anthony Steed, and Martin Usoh. “The Virtual Treadmill: A Naturalistic Metaphor for Navigation in Immersive Virtual Environments”. In: *Proceedings of the Eurographics Workshops on Virtual Environments*. 1995, pp. 135–148.

[166] Mel Slater, Martin Usoh, and Anthony Steed. “Taking Steps: The Influence of a Walking Technique on Presence in Virtual Reality”. In: *ACM Transactions on Computer-Human Interaction* 2.3 (1995), pp. 201–219.

[167] Mario Soilán, Ana Sánchez-Rodrıguez, Pablo del Rıo-Barral, Carlos Perez-Collazo, Pedro Arias, and Belén Riveiro. “Review of Laser Scanning Technologies and Their Applications for Road and Railway Infrastructure Monitoring”. In: *Infrastructures* 4.4 (2019), p. 58.

[168] Henry Sonnet, Sheelagh Carpendale, and Thomas Strothotte. “Integrating Expanding Annotations with a 3D Explosion Probe”. In: *Proceedings of the working conference on Advanced Visual Interfaces*. 2004, pp. 63–70.

[169] Vladeta Stojanovic, Matthias Trapp, Rico Richter, Benjamin Hagedorn, and Jürgen Döllner. “Towards the Generation of Digital Twins for Facility Management Based on 3D Point Clouds”. In: *Proceeding of the 34th Annual ARCOM Conference*. 2018, pp. 270–279.

[170] Qi Sun, Anjul Patney, Li-Yi Wei, Omer Shapira, Jingwan Lun, Paul Asente, Suwen Zhu, Morgan McGuire, David Luebke, and Arie Kaufman. “Towards Virtual Reality Infinite Walking: Dynamic Saccadic Redirection”. In: *Proceedings of ACM SIGGRAPH*. 2018.

[171] Fei Tao, Fangyuan Sui, Ang Liu, Qinglin Qi, Meng Zhang, Boyang Song, Zirong Guo, Stephen C-Y Lu, and AYC Nee. “Digital Twin-Driven Product Design Framework”. In: *International Journal of Production Research* 57.12 (2019), pp. 3935–3953.

[172] Tee-Ann Teo and Chi-Min Chiu. “Pole-Like Road Object Detection from Mobile LiDAR System using a Coarse-to-Fine Approach”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.10 (2015), pp. 4805–4818.

[173] Matthias Trapp, Tassilo Glander, Henrik Buchholz, and Jürgen Döllner. “3D Generalization Lenses for Interactive Focus+Context Visualization of Virtual City Models”. In: *Proceedings of the 12th International Conference on Information Visualisation*. IEEE. 2008, pp. 356–361.

[174] Martin Usoh, Kevin Arthur, Mary C. Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P Brooks Jr. “Walking > Walking-in-place > Flying, in Virtual Environments”. In: *Proceedings of ACM SIGGRAPH*. 1999, pp. 359–364.

[175] Mikael Vaaraniemi, Martin Freidank, and Rüdiger Westermann. “Enhancing the Visibility of Labels in 3D Navigation Maps”. In: *Lecture Notes in Geoinformation and Cartography*. 2012, pp. 23–40.

[176] Juho-Pekka Virtanen, Antero Kukko, Harri Kaartinen, Anttoni Jaakkola, Tuomas Turppa, Hannu Hyyppä, and Juha Hyyppä. “Nationwide Point Cloud – The Future Topographic Core Data”. In: *ISPRS International Journal of Geo-Information* 6.8 (2017), p. 243.

[177] Alex Vlachos. “Advanced VR Rendering”. In: *Presentation at Game Developers Conference*. 2015.

[178] Alex Vlachos. “Advanced VR Rendering Performance”. In: *Presentation at Game Developers Conference*. 2016.

[179] Maximilian Vogt, Adrian Rips, and Claus Emmelmann. “Comparison of iPad Pro®’s LiDAR and TrueDepth Capabilities with an Industrial 3D Scanning Solution”. In: *Technologies* 9.2 (2021), p. 25.

[180] Jillian Walliss and Heike Rahmann. *Landscape Architecture and Digital Technologies: Re-Conceptualising Design and Making*. Routledge, 2016.

[181] Michael Wand, Alexander Berner, Martin Bokeloh, Philipp Jenke, Arno Fleck, Mark Hoffmann, Benjamin Maier, Dirk Staneker, Andreas Schilling, and Hans-Peter Seidel. "Processing and Interactive Editing of Huge Point Clouds from 3D Scanners". In: *Computers & Graphics* 32.2 (2008), pp. 204–220.

[182] Lujin Wang, Ye Zhao, Klaus Mueller, and Arie Kaufman. "The Magic Volume Lens: An Interactive Focus+Context Technique for Volume Rendering". In: *Proceedings of the IEEE Conference on Visualization*. 2005, pp. 367–374.

[183] Qian Wang and Min-Koo Kim. "Applications of 3D Point Cloud Data in the Construction Industry: A Fifteen-Year Review from 2004 to 2018". In: *Advanced Engineering Informatics* 39 (2019), pp. 306–319.

[184] Stefan Weinzierl, Paolo Sanvito, Frank Schultz, and Clemens Büttner. "The Acoustics of Renaissance Theatres in Italy". In: *Acta Acustica united with Acustica* 101.3 (2015), pp. 632–641.

[185] Stephen Wessels, Heinz Ruther, Roshan Bhurtha, and Ralph Schroeder. "Design and Creation of a 3D Virtual Tour of the World Heritage Site of Petra, Jordan". In: *Proceedings of the AfricaGeo conference* (2014), pp. 1–3.

[186] Michael Wimmer and Claus Scheiblauer. "Instant points: Fast Rendering of Unprocessed Point Clouds". In: *Eurographics Symposium on Point-Based Graphics*. 2006, pp. 129–137.

[187] Matthias M Wloka and Eliot Greenfield. "The Virtual Tricorder: A Uniform Interface for Virtual Reality". In: *Proceedings of the 8th annual ACM Symposium on User interface and Software Technology*. 1995, pp. 39–40.

[188] Jainhua Wu and Leif Kobbelt. "Optimized Sub-Sampling of Point Sets for Surface Splatting". In: *Computer Graphics Forum* 23.3 (2004), pp. 643–652.

[189] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, *et al.* "SAPIEN: A SimulAted Part-Based Interactive ENvironment". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11097–11107.

[190] Xuehan Xiong, Antonio Adan, Burcu Akinci, and Daniel Huber. "Automatic Creation of Semantically Rich 3D Building Models from Laser Scanner Data". In: *Automation in Construction* 31 (2013), pp. 325–337.

[191] Hui Xu, Minh X. Nguyen, Xiaoru Yuan, and Baoquan Chen. "Interactive Silhouette Rendering for Point-Based Models". In: *Proceedings of the Eurographics Symposium on Point-Based Graphics* (2004), pp. 13–18.

[192] Zhihang Yao, Claus Nagel, Felix Kunde, György Hudra, Philipp Willkomm, Andreas Donaubauer, Thomas Adolphi, and Thomas H Kolbe. "3DCityDB - a 3D Geodatabase Solution for the Management, Analysis, and Visualization of Semantic 3D City Models Based on CityGML". In: *Open Geospatial Data, Software and Standards* 3.1 (2018), pp. 1–26.

[193] Sarah Younan and Cathy Treadaway. "Digital 3D models of Heritage Artefacts: Towards a Digital Dream Space". In: *Digital Applications in Archaeology and Cultural Heritage* 2.4 (2015), pp. 240–247.

[194] Jihun Yu and Greg Turk. "Reconstructing Surfaces of Particle-Based Fluids using Anisotropic Kernels". In: *ACM Transactions on Graphics* 32.1 (2013), 5:1–5:12.

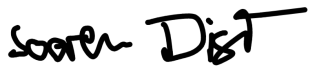
[195] Qian-Yi Zhou and Ulrich Neumann. "2.5D Building Modeling by Discovering Global Regularities". In: *Computer Vision and Pattern Recognition*. 2012, pp. 326–333.

[196] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. "Surface Splatting". In: *Proceedings of ACM SIGGRAPH*. 2001, pp. 371–378.

Statutory Declaration

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Potsdam, December 6, 2022
(Place, Date)


(Signature)