Service-Oriented Processing and Analysis of Massive Point Clouds in Geoinformation Management

Sören Discher, Rico Richter, Matthias Trapp, Jürgen Döllner

Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany

Abstract. Today, landscapes, cities, and infrastructure networks are commonly captured at regular intervals using LiDAR or image-based remote sensing technologies. The resulting point clouds, representing digital snapshots of the reality, are used for a growing number of applications, such as urban development, environmental monitoring, and disaster management. Multi-temporal point clouds, i.e., *4D point clouds*, result from scanning the same site at different points in time and open up new ways to automate common geoinformation management workflows, e.g., updating and maintaining existing geodata such as models of terrain, infrastructure, building, and vegetation. However, existing GIS are often limited by processing strategies and storage capabilities that generally do not scale for massive point clouds containing several terabytes of data. We demonstrate and discuss techniques to manage, process, analyze, and provide large-scale, distributed 4D point clouds. All techniques have been implemented in a system that follows service-oriented design principles, thus, maximizing its interoperability and allowing for a seamless integration into existing workflows and systems. A modular service-oriented processing pipeline is presented that uses out-of-core and GPU-based processing approaches to efficiently handle massive 4D point clouds and to reduce processing times significantly. With respect to the provision of analysis results, we present web-based visualization techniques that apply real-time rendering algorithms and suitable interaction metaphors. Hence, users can explore, inspect, and analyze arbitrary large and dense point clouds. The approach is evaluated based on several real-world applications and datasets featuring different densities and characteristics. Results show that it enables the management, processing, analysis, and distribution of massive 4D point clouds as required by a growing number of applications and systems.

Keywords. 4D point clouds, analytics, visualization, service-oriented infrastructure

1. Introduction and Problem Statement

Over the last decades, 3D point clouds, i.e., discrete, digital representations of 3D objects and environments based on unstructured collections of 3D geometric points, have become an essential data category for geospatial and non-geospatial applications in diverse areas such as building information modeling (Pătrăucean et al. 2015, Stojanovic et al. 2017), urban planning and development (Musialski et al. 2013), or the digital preservation of cultural and natural heritage (Rüther et al. 2012, Hämmerle et al. 2014). A major cause for that popularity surge have been technological advances in remote and in-situ sensing technology, allowing us to capture assets, buildings, cities, or complete countries with unprecedented speed and precision. As an example, state-of-the-art laser scanners may capture millions of points per second, generating highly detailed point clouds of individual sites (e.g., several points per square centimeter) within few hours (Rüther et al. 2012). Larger areas can be covered by attaching the scanning device to a moving vehicle, such as cars or unmanned aircraft systems (UAS), resulting in massive point clouds that may contain several billion points and terabytes of raw data (Leberl et al. 2010, Martinez-Rubi et al. 2015). 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, such as aerial laser scanning and mobile mapping (Nebiker et al. 2010, Puente et al. 2013). The resulting dense and multi-temporal datasets, commonly referred to as *4D point clouds*, may be visualized and used as interactive models that document how a given site or land-scape has changed over time (Discher et al. 2017). Furthermore, they allow us to automate and speed up common geoinformation management workflows: in urban planning and development for example, existing geodata such as official tree cadasters or terrain models, can be efficiently updated based on 4D point clouds (Oehlke et al. 2015). In the wake of natural disasters, remote sensing can be applied to quickly gather up-to-date information about affected areas and to conduct an automated damage assessment based on a comparison to previous scans (Richter et al. 2013b).

However, existing geoinformation systems (GIS) are often limited by processing and visualization techniques that do not scale for such massive datasets. To handle limited processing and memory capabilities (i.e., main and GPU memory), data quality is often reduced, either by thinning the respective point clouds (Peters & Ledoux 2016) or by converting them into generalized 3D meshes (Berger et al. 2014). To overcome these limitations and to make use of the full potential and resolution of 4D point clouds, external memory algorithms are required, which dynamically fetch and unload subsets of the data based on their relevance to the task at hand, limiting the memory usage to a predefined budget. An efficient access to the relevant subsets can be ensured by organizing point clouds using appropriate spatial data structures and level-of-detail (LoD) concepts (Elseberg et al. 2012, Goswami et al. 2013). With data sources becoming more numerous and diverse, an ever-increasing number of stakeholders requires access to the data. Hence, there is a growing demand for a seamless integration of point-based processing and rendering functions into existing workflows and systems. As a result, the efficient and scalable management of 4D point clouds becomes an increasingly relevant aspect (van Oosterom et al. 2015, Cura et al. 2017). Available research solutions use service-oriented design principles to implement scalable systems. As an example, Martinez-Rubi et al. (2015) describe a system that integrates massive point cloud data and makes it publicly accessible via a web interface, allowing users to interactively explore or download arbitrary subsets of the data. Richter & Döllner (2014) propose a system for the management of multi-temporal point clouds, addressing both integration, processing, and rendering functionality. They discuss a service-oriented architecture that focuses on a seamless integration of system components into existing workflows: each component can be accessed individually via standardized web services.

We extend the architecture proposed in Richter & Döllner (2014) by improving processing performance and interoperability. We introduce a *modular processing pipeline* that implements parallel processing strategies to speed up individual tasks and facilitate the distribution of those tasks alongside corresponding datasets within a distributed infrastructure for data storage. The use of external memory algorithms allows us to handle arbitrary large 4D point clouds. In addition, existing web services for geodata can be seamlessly integrated into the processing pipeline to improve processing results. The article is structured as follows: Section 2 discusses the acquisition and management of massive, heterogeneous 4D point clouds, focusing on the characteristics of different data sources and consequential system requirements. The pipeline architecture is presented in Section 3. Section 4 discusses web-based rendering techniques and interaction metaphors that facilitate the exploration and inspection of resulting datasets. In Section 5, we present case studies based on real-world applications and datasets. Section 6 gives conclusions and an outlook on future challenges.

2. Data Acquisition and System Requirements

Systems for the acquisition of point clouds are manifold and allow us to capture real-world surfaces at all scales, ranging from small objects over individual buildings to complete cities or even countries. In this section, we describe the characteristics of common, established acquisition systems and identify requirements for systems aiming to enable the efficient management, processing, and distribution of the resulting data on a massive scale as vital for a variety of applications. Based on these requirements, a corresponding system architecture is presented.

2.1. Data Sources and Characteristics

Today, real-world surfaces are captured using active or passive sensors. Active sensors (e.g., LiDAR, radar, stripe projection systems) emit electromagnetic radiation to directly measure the distance between surface and sensor, generating so-called range data (Eitel et al. 2016). Passive sensors (e.g., aerial cameras, digital cameras, hyperspectral radiometers) on the other hand (Remondino, Spera et al. 2013) solely rely on natural radiation, most notably sunlight, and generate series of images used as input for dense image matching algorithms to derive 3D information (Rothermel et al. 2012). Both, active and passive sensors, can capture individual objects with point densities of up to a few micrometers. The resolution depends on the distance between sensor and captured surface. On a larger scale, point clouds of entire rooms, buildings, or facilities can be efficiently generated by placing sensors at key positions within the site in question (Remondino, Menna et al. 2013). By attaching the sensors to moving vehicles such as cars, trains, UAS, or satellites, data for even larger areas such as infrastructure networks, cities, or countries can be efficiently collected, although at reduced point densities (Ostrowski et al. 2014).

Each acquisition system comes with specific advantages and disadvantages, affecting its suitability for different use cases. 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 (Kersten et al. 2016) and UAS. Furthermore, image-based methods allow for potentially higher point densities, e.g., up to 100 points/m^2 for aerial photographs as in contrast to typically 1-25 points/m^2 for aerial laser scans (Remondino, Spera et al. 2013). 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 (Remondino, Spera et al. 2013). 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 (Remondino, Menna et al. 2013). In practice, both sensor categories are frequently used in parallel. Advanced driver assistance systems for example, combine varying active and passive sensors to observe a car's immediate surroundings (Langner et al. 2016). Table 1 provides an overview of several common acquisition systems and their specific characteristics.

Acquisition System	Typical Scale	Typical Density (pts/m^2)	Costs
Airborne Laserscanning	Infrastructure networks, urban + rural areas	1 - 25	very high
Aerial Photography	Infrastructure networks, urban + rural areas	25 - 100	high
Mobile Mapping (rails, roads)	Infrastructure networks, urban areas	200 - 1,400	medium
UAS	Buildings, facilities, infrastructure networks	500 - 6,000	medium
Static Terrestrial Laserscanning	Indoor scenes, buildings, facilities	4,000 - 20,000	medium
Smartphone cameras / DSLRs	Individual objects, indoor scenes, buildings	4,000 - 40,000	low
Depth Cameras	Individual objects, indoor scenes	4,000 - 20,000	low
Stripe-projection Systems	Individual objects, indoor scenes	100,000 - 400,000	low

Table 1. Commonly used acquisition systems for point clouds and their characteristics.

2.2. Challenges and System Requirements

Traditionally, point clouds are captured, processed, analyzed, and visualized in the scope of only one specific application. However, each dataset might also contain relevant information for completely different use cases. For example, point clouds generated by advanced driver assistance systems, can be of immense value for urban planning and development, as they provide up-to-date information about the infrastructure of a city on a frequent basis and from an often supplementary perspective (i.e., from a pedestrian's perspective instead of a bird's-eye view). Similarly, a frequent scanning of a site raises valuable insights about its history, which has the potential to greatly benefit common geoinformation management workflows such as predictive maintenance and the automated updating of official cadaster data (Section 5). Nonetheless, traditional GIS still operate primarily on 2D or 3D meshes (Musliman et al. 2008, van Oosterom et al. 2008) derived from point clouds in a time-consuming process (Berger et al. 2014). A direct processing of point clouds would be more efficient because point clouds have no limitations regarding model, geometry, structure, or topology and can be updated automatically in contrast to meshes (Paredes et al. 2012). Hence, the use of point-based instead of mesh-based models within geoinformation management workflows has the potential to speed up common tasks and applications notably (Richter & Döllner 2014).

Establishing 4D point clouds as a basic data type to provide geographic content requires systems that facilitate the efficient management of such datasets. In particular, this refers to the integration of heterogeneous point clouds from various data sources, the updating, processing, and analysis of massive datasets, as well as the provision and visualization of arbitrary subsets based on varying per-point attributes (e.g., spatial, temporal, or semantic information). To ensure an efficient access to the data, suitable spatial data structures and LoD concepts are required. Parallel processing strategies need to be implemented and combined with a distributed storage of the point data to significantly improve the performance of a system. To facilitate the exploration of analysis results, suitable visualization techniques and interaction metaphors must be applied that enhance the recognition of objects, semantics, and temporal changes within point cloud depictions.

To ease the integration of such systems into existing workflows and process chains, their interoperability must be guaranteed by making their functionality available via standardized web services (van Oosterom et al. 2008), likewise the integration of existing web services for geodata into the system should be supported. In summary, the following requirements need to be addressed:

R1. Integration of point clouds from heterogeneous data sources into a homogeneous spatial data model,

R2. distributed storage of point clouds,

R3. processing services for the distributed, scalable, adaptive, and selective updating, processing, and analysis of massive 4D point clouds,

R4. support to integrate existing web services for geodata into the system to further increase interoperability,

R5. visualization services to provide 4D point clouds for heterogeneous clients (e.g., desktop computers, mobile devices),

R6. visualization and interaction techniques to enable a task-specific and application-specific visualization of massive 4D point clouds.

The next section presents a system architecture that fulfills these requirements.

2.3. System Architecture

We propose a service-oriented architecture that builds upon the system proposed by Richter & Döllner (2014), but further improves its scalability and interoperability (Figure 1). Generally, managing 4D point clouds comprises three stages:

1) **Data Integration.** Components that implement the integration of point clouds from heterogeneous data sources into a homogeneous spatial data model (R1). This includes georeferencing of acquired data as well as preparing or updating LoD data structures to ensure an efficient data access in subsequent stages. Another aspect is the filtering and quality control of the data (e.g., thinning, noise reduction, and outlier filtering).
2) **Data Analytics.** Components for common and domain-specific analyses and simulations (R3). Those typically require additional per-point attributes (e.g., normal, topological, or surface category information) that are either computed by specific preprocessing components or provided by external web services, which can be seamlessly integrated into the system (R4). Depending on the use case, analysis results can be stored as additional per-point attributes or exported as 2D or 3D meshes (e.g., shapes).
3) **Data Provision.** Components providing access to data and analysis results for external processing and visualization tools. The data is either provided in standardized geodata formats via Web Feature Services (Rautenbach et al. 2013) or optimized for a web-based visualization using Web 3D Services or Web View Services (Klimke et al. 2013). This allows for the interactive exploration of the data on heterogeneous client systems (R5), ranging from high-end workstations to low-end mobile devices with very limited computing capabilities, network bandwidth, or memory resources.

The individual components are chained together in a modular processing pipeline that implements parallel and distributed computing concepts to allow for the efficient and scalable execution of updating, processing, and analysis tasks (R3). The processing pipeline can be accessed and reconfigured via a Web Processing Service (Mueller & Pross 2015) to dynamically adapt the applied analyses and simulations to the field of application and current use cases (Section 3). Our system stores point clouds in a distributed way: If required, additional computing and storage resources may be added at runtime, maximizing its scalability and availability (R2). A standardized interface to integrate, update, and access arbitrary subsets of the data is provided by a central system component, which we refer to as *point cloud database*. Spatial data structures, that hierarchically subdivide the spatial area, and LoD concepts, that provide subset representations, are used by that database to reduce the amount of data adaptively for data queries and processing tasks. The suitability of different *LoD data structures* (Richter & Döllner 2010, Elseberg et al. 2012, Goswami et al. 2013) varies based on the application scenario (e.g., real-time visualization or analysis) and spatial distribution of the data (e.g., terrestrial or airborne). Evaluating, generating, and maintaining them is also the responsibility of the point cloud database.

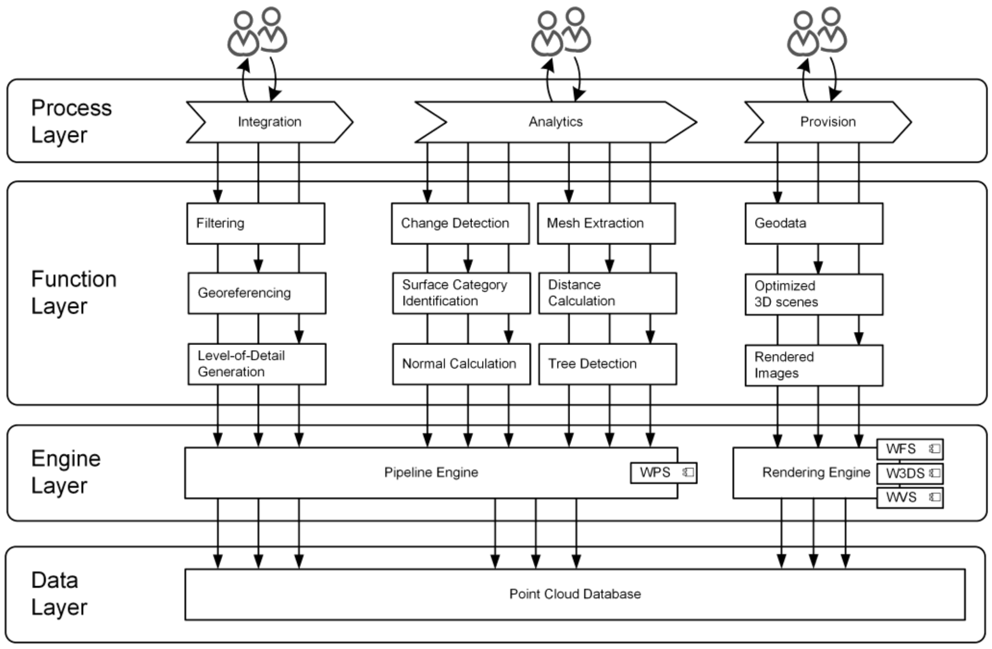

Figure 1. Service-oriented architecture of our system with arrows indicating usage relations between components.

3. Service-oriented Point Cloud Analytics

Processing or analyzing 4D point clouds requires typically only a small subset of the entire data set at the same time. Typical tasks and complex computations that must be computed for each point in a dataset such as duplicate detection, change detection, and classification operate on a small proximity around each point (Richter et al. 2013a, Belgiu et al. 2014). Hence, the processing performance can be significantly increased by applying parallel computing concepts, either based on a CPU or GPU. Furthermore, workflows that include multiple 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. By splitting the data into sufficiently small subsets, even massive point clouds, which exceed available memory capacities, can be handled efficiently. In this section, we present a modular pipeline architecture (Figure 2) that implements those concepts: 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 within networks (R3). Each analysis is described by a processing pipeline that defines involved processing tasks and can be reconfigured and replaced dynamically. Also, the pipeline architecture can be easily expanded to integrate existing web services for geodata, thus, maximizing its interoperability (R4).

3.1. Pipeline Architecture

The proposed architecture 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 3.2); second, a pipeline engine to configure and execute various processing pipelines. Each pipeline defines a specific combination of basic input, processing and output tasks. We define the elements that compose a pipeline as follows:

- **Importers**, i.e., pipeline nodes that import data from any source such as files, the *point cloud database* or other, external sources (e.g., web-services). 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/LAZ), the point cloud database or other geodata (e.g., shape files, CityGML, GeoTIFFs). The latter functionality makes exporters essential in facilitating the integration of the proposed system into existing workflows.
- **Tasks**, i.e., pipeline nodes that implement a specific processing or analysis algorithm. Some algorithms operate on multiple data sets 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 defined per-point attributes that are required.
- **Data Packages**, i.e., data subsets that are transferred between pipeline nodes via connections. Similar to *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. The pipeline engine allows to execute multiple pipeline plans in parallel, each of which can be started, paused, and stopped dynamically. At runtime, each active pipeline node gets assigned its own set of resources by the resource manager. Processed data packages are immediately transferred to subsequent pipeline nodes. For each incoming connection, a pipeline node manages a queue of incoming data packages. The query size is limited to a defined number of data packages. If a queue reaches its maximum capacity, no additional data packages are accepted and preceding nodes are not executed. 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).

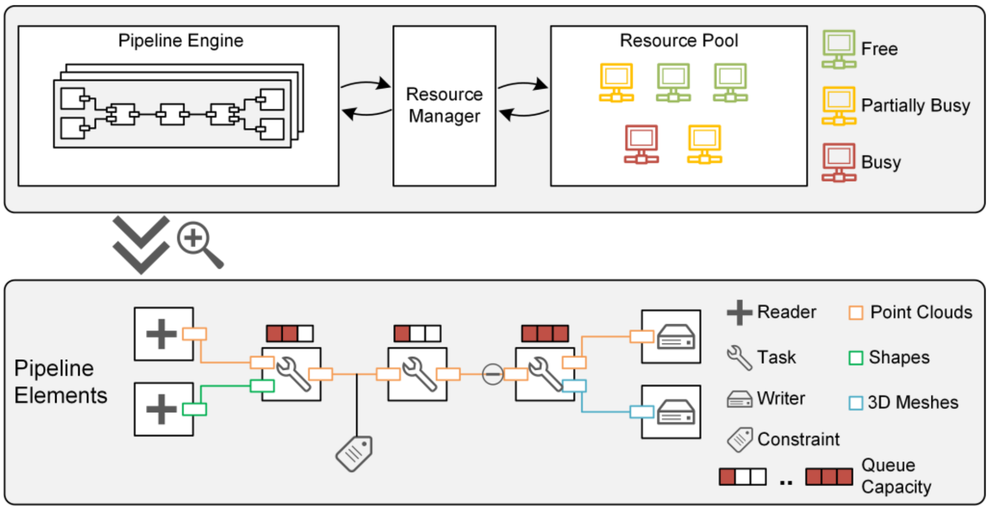

Figure 2. Overview of the pipeline architecture and pipeline elements.

3.2. 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 of the system. Whenever a pipeline node needs to be executed, the resource manager assigns resources based on available system capabilities. After the execution is finished, all assigned resources are released to the resource pool and become available for other nodes (Figure 2). 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 nodes.
- **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 nodes may vary significantly depending on the task. An adaptive resource scheduling allows to prevent bottlenecks in the processing. The execution time is tracked for each node and the number of assigned resources is adjusted dynamically.

4. Point Cloud Visualization

The interactive exploration of massive 4D point clouds is an essential functionality of our system, facilitating the visual recognition and interpretation of objects, semantics, and analysis results within corresponding data sets. In this section, we discuss web-based rendering approaches allowing to render arbitrary large data sets on a multitude of heterogeneous clients (R5, Section 4.1). Furthermore, we describe how visual filtering and highlighting within point cloud depictions can be facilitated by combining different visualization techniques and interaction metaphors (R6, Section 4.2).

4.1. Web-based Rendering

Since point clouds with billions of points exceed typically available main and GPU memory capacities by an order of magnitude, out-of-core rendering techniques are required to decouple rendering efforts from the amount of data. Exemplary systems (Goswami et al. 2013, Richter et al. 2015) use LoD data structures to organize the data into chunks that are suitable for fast processing and rendering. The LoD node selection is performed on a per-frame basis and depends on the view position and user interaction, as well as computing and storage capabilities of the underlying rendering system. Different caching approaches are used to minimize data swapping latencies for memory transfers between secondary storage, main memory, and GPU memory.

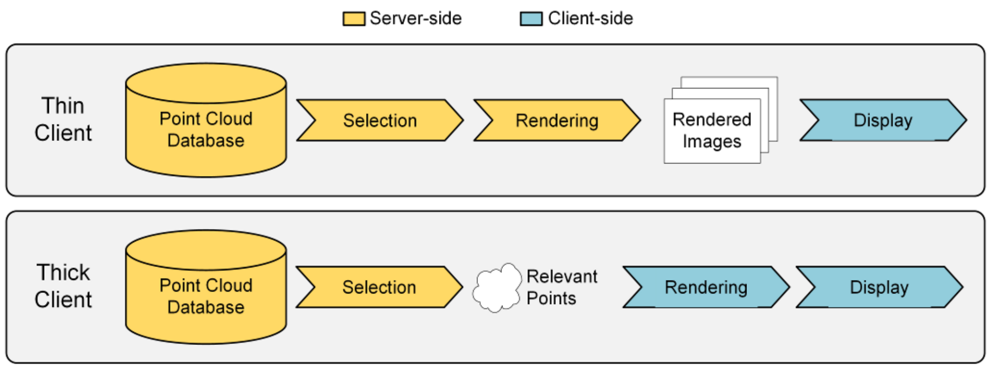

Figure 3. Service-oriented rendering approaches: thin clients vs. thick clients.

While being applicable to arbitrary large data sets, out-of-core rendering algorithms require a direct access to the data, which generally restricts their application to systems with massive storage capacities. For other systems such as mobile devices, these algorithms must be combined with existing web-based approaches that limit workload and data traffic on client-side by using a central server infrastructure to maintain and distribute the data (R5, Figure 3). Rendering directly on the server and only transferring the rendered images to the client is commonly referred to as a thin client approach (Döllner et al. 2012, Klimke et al. 2014). As an optimization, many of such approaches render and transfer cube-maps instead of individual images. Thus, new data requests are only required when the view position changes significantly. Alternatively, thick client approaches can be used that delegate the rendering to the client side. Here, the server is only responsible for selecting and transferring the data to the client (Krämer & Gutbell 2015, Martinez-Rubi et al. 2015, Limberger et al. 2016b, Schoedon et al. 2016). While a thin client approach notably reduces the minimal hardware requirements on client side, a thick client approach is usually more feasible to serve a large number of clients due to lower workload generated on server side. Furthermore, a thick client approach tends to be more resilient to unstable network connections since the visualization can still be adjusted to user interactions when the connection to the server has been lost temporarily, albeit some relevant data might be missing.

4.2. Semantic-based Visualization

The visualization of point clouds can be adapted to various objectives (e.g., highlighting structural changes within multi-temporal datasets) by switching between different point-based rendering techniques and color schemes

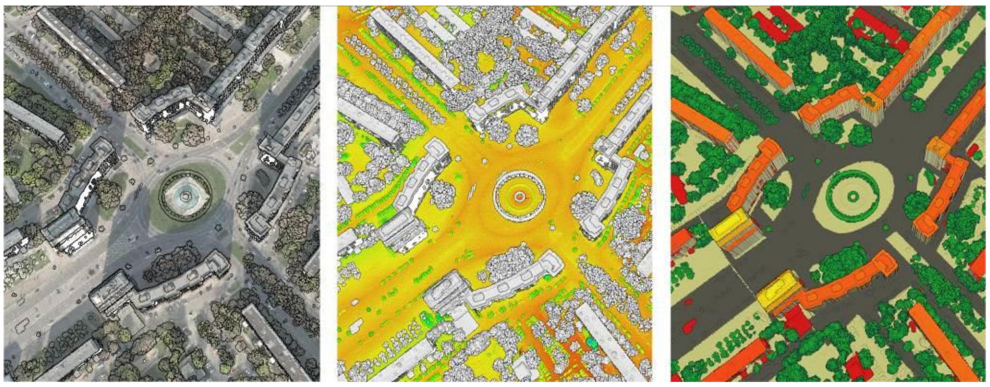

Figure 4. Visualization of an airborne laser scan of an urban area, showcasing different rendering setups using per-point surface category and topological information: (left) Uniform rendering for all categories using rgb colors. (middle) Height gradient applied to ground points, uniform color and edge highlighting applied to other categories. (right) Height gradient and closed surface rendering applied to building points, category-specific colors used for ground, street and vegetation points.

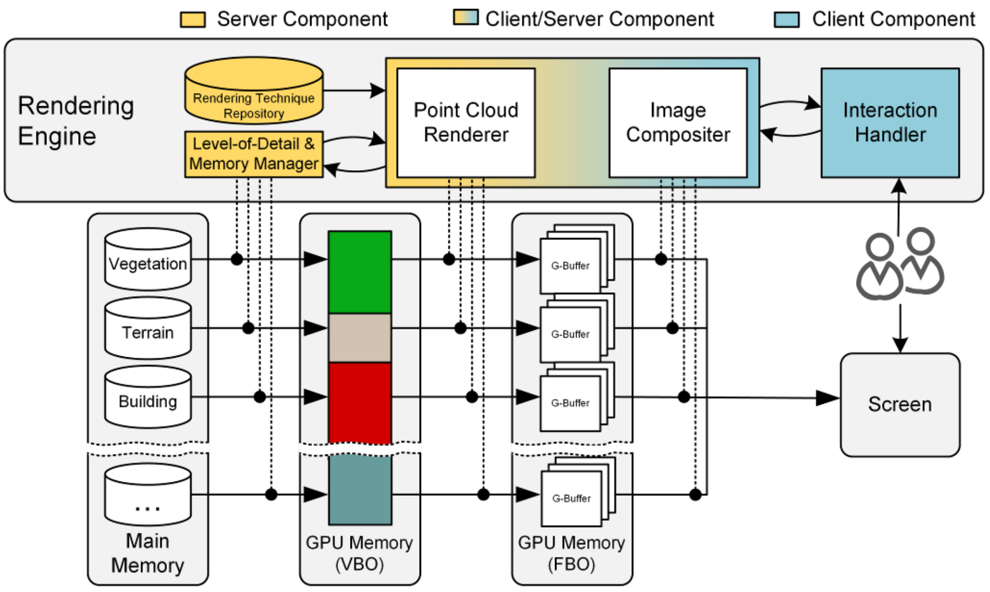

Figure 5. Overview of the proposed rendering engine. Prior to rendering, points are separated based on per-point attributes (e.g., surface categories).

(Gross & Pfister 2007). Many rendering techniques focus on a photorealistic visualization: points are represented as splats, i.e., oriented disks, spheres, or paraboloids. An adequate size and orientation enables to render a closed surface (Preiner et al. 2012, Schütz & Wimmer 2015). However, a

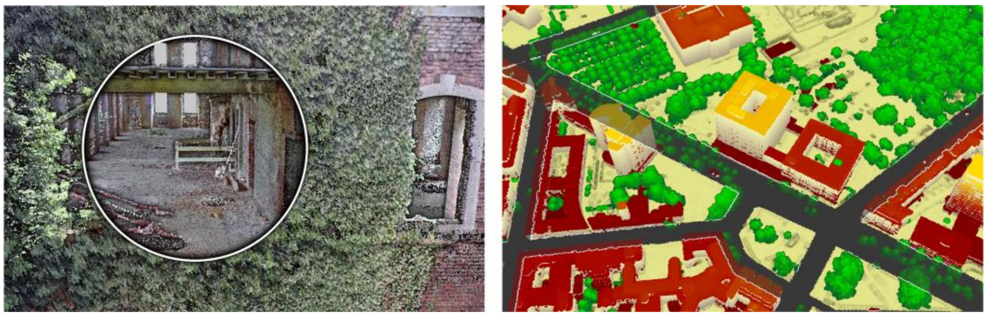
Figure 6. Focus+context visualization for massive point clouds. Left: Using an interactive see-through lens, the occluded interior of a building can be inspected in the context of the overall scan. Right: A visibility mask is used to highlight otherwise hidden street points.

photorealistic representation often hinders the visual identification and categorization of structures and topological properties. In airborne datasets for example, vegetation can be difficult to distinguish from ground points and small structures might not be visible due to an insufficient contrast to the environment (Richter et al. 2015). Meanwhile, non-photorealistic rendering techniques (Boucheny 2009) efficiently highlight edges and structures within point cloud depictions.

In general, task-specific explorations of virtual 3D scenes can be facilitated by combining different rendering techniques, color schemes, and post-processing effects (Döllner et al. 2005, Semmo et al. 2015, Würfel et al. 2015, Limberger et al. 2016a). For point cloud depictions, this is exemplified by Richter et al. (2015) who use per-point attributes (e.g., surface category, topologic information) to adapt the appearance of a point, i.e., its color, size, orientation, or shape (Figure 4). They apply multi-pass rendering utilizing G-Buffers for image-based compositing (Figure 5). Points are grouped into different subsets (e.g., based on their surface category), each of which is rendered separately. A compositing pass merges the rendering results, enabling sophisticated focus+context visualization and interaction techniques (Elmqvist & Tsigas 2008, Vaaraniemi et al. 2013, Semmo & Döllner 2014) such as visibility masks (Sigg et al. 2012) or interactive lenses (Trapp et al. 2008, Pasewaldt et al. 2012). Thus, task-relevant structures can be easily identified even if they are fully or partly occluded (R6, Figure 6). All rendering and interaction techniques can be selected, configured, and combined at run-time.

5. Case Studies

We implemented the proposed *pipeline engine* using C++, CUDA (Storti & Yurtoglu 2015), and the Point Cloud Library (Rusu & Cousins 2011) as basic

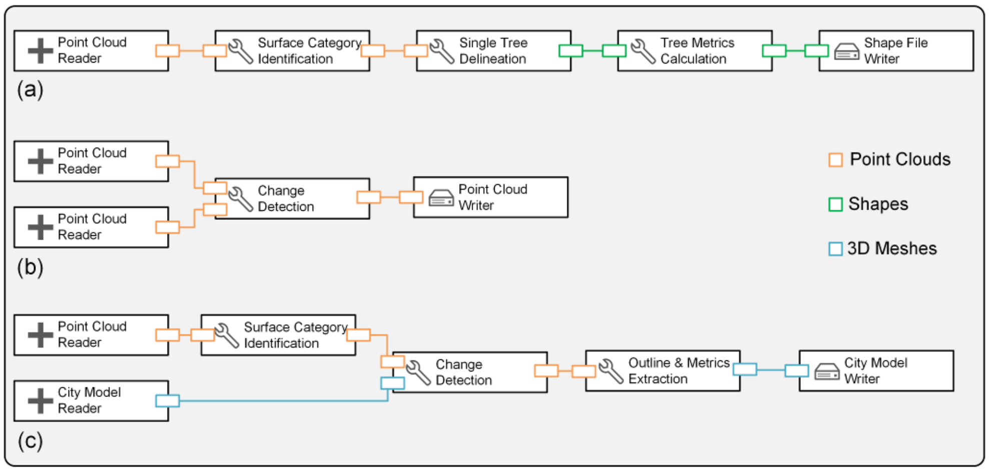

Figure 7. Exemplary processing pipelines as they have been used for the case studies.

technologies. For the *rendering engine*, we additionally used OpenGL (Shreiner et al. 2013) and OpenSceneGraph (Wang & Qian 2010) for the thin client approach as well as WebGL and Cesium (Krämer & Gutbell 2015) for the thick client approach. Test data sets were integrated into a layered LoD data structure with each layer representing a different surface category (e.g., ground, vegetation, buildings) and acquisition date. The LoD data structure and corresponding data chunks were serialized to files acting as a *database*. All tests and measurements were performed by a small network containing a total of six 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. As case studies, we evaluated the following scenarios:

Deriving Tree Cadasters. Tree cadasters consolidate detailed information about biomass in an area and are essential for a growing number of applications in 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 (Oehlke et al. 2015). As depicted in Figure 7a, the 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 classification (Richter, Behrens & Döllner 2013) was applied that distinguishes between ground, building, and vegetation by analyzing for each point the topology of its local neighborhood. To efficiently delineate individual trees, a point-based approach by Li et al (2012) 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 7b) is commonly referred to as a change detection (Kang & Lu 2011, Eitel et al. 2016). 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 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 virtual 3D city model (Figure 7c), indicating newly constructed, removed, 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.

Test data sets for those scenarios comprised airborne, terrestrial, and mobile scans covering different parts of three metropolitan regions with the largest point cloud featuring 100 points/m^2 for an area of 800 km^2. In Table 2, the data throughput for the most relevant processing tasks 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.

Processing Task	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

Table 2. Average data throughput. Change detection was run on airborne, terrestrial, and mobile mapping data, all other tasks were performed for airborne data sets.

6. Conclusions and Future Work

By applying out-of-core, parallel, and distributed processing strategies, 4D point clouds of any size can be updated and analyzed at significantly reduced processing times. Our modular pipeline architecture implements such concepts, making efficient use of available hardware resources. A database component is used for the distributed storage and efficient integration of point clouds from heterogeneous data sources, ensuring efficient access to arbitrary subsets of the data. A rendering component allows for analysis results to be visualized and inspected on heterogeneous clients with differing computing capabilities by implementing web-based and out-of-core rendering algorithms. To facilitate the inspection, specialized visualization and interaction techniques are provided. All system components can be configured at runtime, allowing to design task and domain-specific analysis and inspection tools. They also provide interoperability to existing workflows and process chains by following service-oriented design principles. A prototypical implementation of our system has been successfully tested for several real-world scenarios. Novel devices and sensors have the capability to generate point clouds in real-time, resulting in massively redundant point cloud streams. Hence, ad-hoc data management, analysis, and visualization tools are fundamental requirements. Our future work will focus on challenges introduced by such point cloud streams to use them even for applications which are not in the core area of GIS to expedite the digital transformation.

Acknowledgements

This work was funded by the Federal Ministry of Education and Research (BMBF), Germany within the InnoProfile Transfer research group “4DnD-Vis” and the Research School on “Service-Oriented Systems Engineering” of the Hasso-Plattner-Institut. We would like to thank virtualcitySYSTEMS for providing datasets.

References

Belgiu M, Tomljenovic I, Lampoltshammer TJ, Blaschke T, Höfle B (2014) Ontology-Based Classification of Building Types Detected from Airborne Laser Scanning Data. Remote Sensing, 6(2), pp. 1347-1366

Berger M, Tagliasacchi A, Seversky L, Alliez P, Levine J, Sharf A, Silva C (2014) State of the Art in Surface Reconstruction from Point Clouds. Eurographics STARs, 1(1), pp. 161-185

Boucheny C (2009) Interactive Scientific Visualization of Large Datasets: Towards a Perceptive-Based Approach. Ph.D. Thesis, Université Joseph Fourier, Grenoble, France

Cura R, Perret J, Paparoditis N (2017) A Scalable and Multi-Purpose Point Cloud Server (PCS) for Easier and Faster Point Cloud Data Management and Processing. ISPRS Journal of Photogrammetry and Remote Sensing, 127, pp. 39-56

Discher S, Richter R, Döllner J (2017) Interactive and View-Dependent See-Through Lenses for Massive 3D Point Clouds. Advances in 3D Geoinformation, pp. 49-62

Döllner J, Buchholz H, Nienhaus M, Kirsch F (2005) Illustrative visualization of 3D city models. Electronic Imaging 2005, pp. 42-51

Döllner J, Hagedorn B, Klimke J (2012) Server-Based Rendering of Large 3D Scenes for Mobile Devices using G-buffer Cube Maps. 17th International Conference on 3D Web Technology, pp. 97-100

Eitel J, Höfle B, Vierling L, Abellán A, Asner G, Deems J, Glennie C, Joerg P, LeWinter A, Magney T, Mandlburger G (2016) Beyond 3-D: The New Spectrum of Lidar Applications for Earth and Ecological Sciences. Remote Sensing of Environment, 186, pp. 372-392

Elmqvist N, Tsigas P (2008) A Taxonomy of 3D Occlusion Management for Visualization. IEEE Transactions on Visualization and Computer Graphics, 14(5), pp. 1095-1109

Elseberg J, Borrmann D, Nüchter A (2013) One Billion Points in the Cloud–an Octree for Efficient Processing of 3D laser scans. ISPRS Journal of Photogrammetry and Remote Sensing, 76, pp. 76-88

Goswami P, Erol F, Mukhi R, Pajarola R, Gobbetti E (2013) An Efficient Multi-Resolution Framework for High Quality Interactive Rendering of Massive Point Clouds using Multi-way kD-Trees. The Visual Computer, 29(1), pp. 69-83

Gross M, Pfister H (2011) Point-Based Graphics (Eds). Morgan Kaufmann.

Hämmerle M, Höfle B, Fuchs J, Schröder-Ritzrau A, Vollweiler N, Frank N (2014) Comparison of Kinect and Terrestrial Lidar Capturing Natural Karst Cave 3-D Objects. IEEE Geoscience and Remote Sensing Letters, 11(11), pp. 1896-1900

Kang Z, Lu Z (2011) The Change Detection of Building Models using Epochs of Terrestrial Point Clouds. International Workshop on Multi-Platform/Multi-Sensor Remote Sensing and Mapping, pp. 1-6

Kersten TP, Przybilla HJ, Lindstaedt M, Tschirschwitz F, Misgaiski-Hass M (2016) Comparative Geometrical Investigations of Hand-Held Scanning Systems. ISPRS Archives, XLI-B5, pp. 507-514

Klimke J, Hagedorn B, Döllner J (2013) A Service-Based Concept for Camera Control in 3D Geovirtual Environments. Progress and New Trends in 3D Geoinformation Sciences, pp. 101-118

Klimke J, Hagedorn B, Döllner J (2014) Scalable Multi-Platform Distribution of Spatial 3D Contents. International Journal of 3-D Information Modeling, 3(3), pp. 35-49

Krämer M, Gutbell R (2015) A Case Study on 3D Geospatial Applications in the Web using State-of-the-Art WebGL Frameworks. 20th International Conference on 3D Web Technology, pp. 189-197

Langner T, Seifert D, Fischer B, Goehring D, Ganjineh T, Rojas R (2016) Traffic Awareness Driver Assistance based on Stereovision, Eye-Tracking, and Head-Up Display. IEEE International Conference on Robotics and Automation, pp. 3167-3173

Li W, Guo Q, Jakubowski M, Kelly M (2012) A new Method for Segmenting Individual Trees from the Lidar Point Cloud. Photogrammetric Engineering & Remote Sensing, 78(1), pp. 75-84

Limberger D, Fiedler C, Hahn S, Trapp M, Döllner J (2016a) Evaluation of Sketchiness as a Visual Variable for 2.5D Treemaps. 20th International Conference Information Visualisation, pp. 183-189

Limberger D, Scheibel W, Lemme S, Döllner J (2016b) Dynamic 2.5D Treemaps using Declarative 3D on the Web. 21st International Conference on Web3D Technology, pp. 33-36

Martinez-Rubi O, Verhoeven S, van Meersbergen M, Schütz M, van Oosterom P, Gonçalves R, Tijssen T (2015) Taming the beast: Free and Open-Source Massive Point Cloud Web Visualization. Capturing Reality Forum 2015

Mueller M, Pross B (2015) OGC WPS 2.0 Interface Standard. Open Geospatial Consortium

Musialski P, Wonka P, Aliaga DG, Wimmer M, Gool LV, Purgathofer W (2013) A Survey of Urban Reconstruction. Computer Graphics Forum, 32(6), pp. 146-177

Musliman IA, Rahman AA, Coors V (2008) Implementing 3D network analysis in 3D GIS. International archives of ISPRS, 37 (part B), pp. 913-918

Nebiker S, Bleisch S, Christen M (2010) Rich Point Clouds in Virtual Globes–A new Paradigm in City Modeling? Computers, Environment and Urban Systems, 34(6), pp. 508-517

Oehlke C, Richter R, Döllner J (2015) Automatic Detection and Large-Scale Visualization of Trees for Digital Landscapes and City Models based on 3D Point Clouds. 16th Conference on Digital Landscape Architecture, pp. 151-160

Ostrowski S, Jozkow G, Toth C, Vander Jagt B (2014) Analysis of Point Cloud Generation from UAS Images. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2(1), pp. 45-51

Pasewaldt S, Semmo A, Trapp M, Döllner, J (2012) Towards Comprehensible Digital 3D Maps. International Symposium on Service-Oriented Mapping, pp. 261–276

Paredes EG, Bóo M, Amor M, Bruguera JD, Döllner J (2012) Extended Hybrid Meshing Algorithm for Multiresolution Terrain Models. International Journal of Geographical Information Science, 26(5), pp. 771-793

Pătrăucean V, Armeni I, Nahangi M, Yeung J, Brilakis I, Haas C (2015) State of Research in Automatic As-Built Modelling. Advanced Engineering Informatics, 29(2), pp. 162-171

Peters R, Ledoux H (2016) Robust approximation of the Medial Axis Transform of LiDAR Point Clouds as a Tool for Visualization. Computers & Geosciences, 90(A), pp. 123-133

Puente I, González-Jorge H, Martínez-Sánchez J, Arias P (2013) Review of Mobile Mapping and Surveying Technologies. Measurement, 46(7), pp. 2127-2145

Preiner R, Jeschke S, Wimmer M (2012) Auto Splats: Dynamic Point Cloud Visualization on the GPU. Eurographics Symposium on Parallel Graphics and Visualization, pp. 139-148

Rautenbach V, Coetzee S, Iwaniak A (2013) Orchestrating OGC Web Services to Produce Thematic Maps in a Spatial Information Infrastructure. Computers, Environment and Urban Systems, 37, pp. 107-120

Remondino F, Menna F, Koutsoudis A, Chamzas C, El-Hakim S (2013) Design and Implement a Reality-Based 3D Digitisation and Modelling Project. Digital Heritage International Congress 2013, pp. 137-147

Remondino F, Spera MG, Nocerino E, Menna F, Nex F, Gonizzi-Barsanti S (2013) Dense Image Matching: Comparisons and Analyses. Digital Heritage Intl. Congress, pp. 47-54

Richter R, Behrens M, Döllner J (2013a) Object Class Segmentation of Massive 3D Point Clouds of Urban Areas using Point Cloud Topology. International Journal of Remote Sensing, 34(23), pp. 8408-8424

Richter R, Discher S, Döllner J (2015) Out-of-Core Visualization of Classified 3D Point Clouds. 3D Geoinformation Science, pp. 227-242

Richter R, Döllner J (2010) Out-of-Core Real-Time Visualization of Massive 3D Point Clouds. 7th International Conference on Virtual Reality, Computer Graphics, Visualization and Interaction in Africa, pp. 121-128

Richter R, Döllner J (2014) Concepts and Techniques for Integration, Analysis and Visualization of Massive 3D Point Clouds. Computers, Environment and Urban Systems, 45, pp. 114-124

Richter R, Kyprianidis JE, Döllner J (2013b) Out-of-Core GPU-based Change Detection in Massive 3D Point Clouds. Transactions in GIS, 17(5), pp. 724-741

Rothermel M, Wenzel K, Fritsch D, Haala N (2012) Sure: Photogrammetric Surface Reconstruction from imagery. LC3D Workshop, Berlin (Vol. 8)

Rusu RB, Cousins S (2011) 3D is Here: Point Cloud Library (pcl). IEEE International Conference on Robotics and automation, pp. 1-4

Rüther H, Held C, Bhurtha R, Schroeder R, Wessels S (2012). From Point Cloud to Textured Model, the Zamani Laser Scanning Pipeline in Heritage Documentation. South African Journal of Geomatics, 1(1), pp. 44-59

Schoedon A, Trapp M, Hollburg H, Döllner J (2016) Interactive Web-based Visualization for Accessibility Mapping of Transportation Networks. EuroVis 2016 – Short Papers

Semmo A, Döllner J (2014) An Interaction Framework for Level-of-Abstraction Visualization of 3D Geovirtual Environments. 2nd ACM SIGSPATIAL International Workshop on Interacting with Maps, pp. 43-49

Semmo A, Trapp M, Jobst M, Döllner J (2015) Cartography-Oriented Design of 3D Geospatial Information Visualization – Overview and Techniques. The Cartographic Journal, 52(2), pp. 95-106

Shreiner D, Sellers G, Kessenich JM, Licea-Kane BM (2013) OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3. Addison-Wesley Professional

Schütz M, Wimmer M (2015) High-Quality Point-Based Rendering using Fast Single-Pass Interpolation. Digital Heritage, pp. 369-372

Sigg S, Fuchs R, Carnecky R, Peikert R (2012) Intelligent Cutaway Illustrations. IEEE Pacific Visualization Symposium, pp. 185-192

Stojanovic V, Richter R, Trapp M, Döllner J (2017) Comparative Visualization of BIM Geometry and Corresponding Point Clouds. International Journal of Sustainable Development and Planning, 13(1), pp. 12-23

Storti D, Yurtoglu M (2015) CUDA for Engineers: An Introduction to High-Performance Parallel Computing. Addison-Wesley Professional

Trapp M, Glander T, Buchholz H, Döllner J (2008) 3D Generalization Lenses for Interactive Focus+Context Visualization of Virtual City Models. 12th International Conference Information Visualisation, pp. 356-361

Vaaraniemi M, Freidank M, Westermann R (2013) Enhancing the Visibility of Labels in 3D Navigation Maps. Progress and new trends in 3D geoinformation sciences, pp. 23-40 van Oosterom P, Martinez-Rubi O, Ivanova M, Horhammer M, Geringer D, Ravada S, Tijssen T, Kodde M, Gonçalves R (2015) Massive Point Cloud Data Management: Design, Implementation and Execution of a Point Cloud Benchmark. Computers & Graphics, 49, pp. 92-125

van Oosterom P, Zlatanova S, Penninga F, Fendel E (2008). Advances in 3D geoinformation systems (Eds). Springer.

Wang R, Qian X (2010) OpenSceneGraph 3.0: Beginner's Guide. Packt Publishing

Würfel H, Trapp M, Limberger D, Döllner J (2015) Natural Phenomena as Metaphors for Visualization of Trend Data in Interactive Software Maps. Computer Graphics and Visual Computing 2015, pp. 69-76