GeoXperience: A Web-Based Platform for Interactive Exploration of Large-Scale Geospatial 3D Data

Udo Feuerhake[1]

Abstract
The increasing volume and complexity of geospatial 3D data demand interactive exploration methods that remain accessible across devices and usage contexts. This paper presents GeoXperience, a web-based platform for immersive exploration of large-scale geospatial 3D datasets directly in the browser. The system supports heterogeneous data representations, including point clouds and voxel-based models, and integrates first-person navigation, virtual reality interaction, and alternative input modalities. A chunk-based data organization combined with view-dependent streaming, adaptive level-of-detail control, and client-side performance management enables interactive exploration under varying hardware and network conditions. Experimental results demonstrate stable frame rates, progressive visual refinement, and consistent interaction behavior across datasets and devices. The platform complements existing web-based and desktop-based visualization approaches by focusing on experience-oriented, installation-free exploration of geospatial 3D data at local scale.

Keywords 3d visualization· streaming· performance · webgl geospatial data adaptive rendering

1 Introduction

The growing complexity and volume of geospatial 3D datasets demand intuitive approaches for interactive exploration. Advances in airborne and terrestrial laser scanning, photogrammetry, and mobile mapping systems have led to rapidly increasing data volumes and geometric complexity. As a result, traditional visualization and interaction concepts are often insufficient for gaining an intuitive spatial understanding, particularly when working with large and heterogeneous datasets.

To address these challenges, this paper presents GeoXperience, a web-based platform designed to support interactive exploration of geospatial 3D data. The platform enables users to move within the data and to switch between different perspectives and interaction modes, including first-person navigation, immersive virtual reality (VR), and gesture-based controls. Rather than treating 3D datasets solely as objects to be inspected from an external viewpoint, the approach emphasizes spatial presence and intuitive interaction.

Several existing systems already provide web-based solutions for 3D geospatial visualization. Potree (Schütz et al., 2020), for example, focuses on the efficient rendering of very large point clouds using hierarchical data structures and out-of-core streaming techniques. CesiumJS (CesiumGS, 2024) enables visualization of global-scale geospatial datasets based on standardized 3D tiling formats and is widely used for city models, terrain representations, and satellite imagery. While these frameworks offer powerful and well-established solutions for specific application scenarios, they primarily focus on visualization aspects and predefined navigation concepts.

Recent research has also investigated browser-based integration of heterogeneous 3D geospatial data using open-source WebGL technologies. La Guardia et al. (2022), for example, present a web-based visualization approach that combines multiple 3D data sources within an interactive browser environment, highlighting the potential of WebGL-based platforms for accessible geospatial exploration. Similarly, Buyukdemircioglu & Kocaman (2018) demonstrate a WebGL-based 3D campus application derived from city models, illustrating the potential of browser-based platforms for local-scale geospatial exploration. Lee & Jang (2019) present an open platform architecture for 3D spatial data visualization in the browser, addressing system design and performance aspects.

More recent web-based frameworks further broaden the scope of browser-based geospatial visualization. For instance, maplibre-gl-lidar extends the MapLibre GL JS

[1] Leibniz University of Hanover, Institute of Cartography and Geoinformatics, Appelstraße 9a, D-30167 Hannover
E-Mail: feuerhake@ikg.uni-hannover.de

rendering engine to support LiDAR point cloud visualization within a vector-tile map context, illustrating efforts to integrate raw 3D data into web-mapping environments. Likewise, Giro3D provides a general WebGL-based framework capable of rendering heterogeneous 2D and 3D geospatial datasets—including raster maps, vector layers, and 3D point clouds—by leveraging three.js for flexible scene management. However, these systems typically emphasize map-centric interaction paradigms or general-purpose 3D rendering and do not explicitly address immersive or first-person exploration concepts for large, heterogeneous datasets.

In addition to web-based approaches, a wide range of established desktop applications exist for processing and inspecting geospatial 3D data. Desktop-based tools such as CloudCompare (CloudCompare Development Team, 2024) and MeshLab (Cignoni et al., 2021) provide extensive analysis and processing functionality and remain indispensable for expert workflows. Similarly, geographic information systems such as ArcGIS Pro (Esri, 2024) and QGIS (QGIS Development Team, 2024) offer integrated environments for point cloud visualization and spatial analysis. These applications, however, require local installation, are often bound to specific operating systems, and typically rely on classical mouse-and-keyboard interaction paradigms. This limits their suitability for lightweight deployment scenarios, interdisciplinary collaboration, or educational and public-outreach contexts, where ease of access and intuitive interaction are of particular importance.

Against this background, GeoXperience focuses on immersive, local-scale exploration directly within the web browser. By relying exclusively on web technologies, the platform can be accessed across different devices, operating systems, and hardware configurations without additional installation, thereby lowering the entry barrier for non-expert users and enabling flexible deployment scenarios.

To achieve interactive performance despite large data volumes, the platform combines spatial chunking, multi-resolution levels of detail, and adaptive streaming strategies. Data preprocessing is intentionally kept lightweight and primarily consists of spatial subdivision and metadata generation, while rendering complexity is adjusted dynamically according to system performance. In addition to visualization efficiency, particular attention is given to interaction concepts. The platform integrates first-person navigation with collision handling, immersive VR via WebXR, and gesture-based controls, drawing on concepts from geospatial visualization, computer graphics, and game-based interaction design.

By combining adaptive data handling with immersive interaction paradigms, GeoXperience aims to complement existing visualization frameworks and extend their capabilities toward experience-oriented exploration of large-scale geospatial 3D data.

The main contributions of this paper are:

1. A general, web-based framework for immersive exploration of geospatial 3D data, supporting multiple data representations and interaction paradigms.
2. An adaptive rendering and streaming concept that enables real-time interaction with large datasets under varying hardware and network conditions.
3. An interaction-oriented design approach that integrates concepts from virtual reality and game-based navigation into geospatial visualization workflows.

The remainder of this paper is structured as follows. Section 2 describes the system architecture and implementation concepts of GeoXperience. Section 3 presents results from performance tests on different devices, which are discussed in Section 4. Finally, Section 5 concludes the paper and outlines directions for future research and development.

2 Methods

2.1 System Overview and Design Principles

GeoXperience is designed as a web-based platform for interactive exploration of large-scale geospatial 3D data. It enables direct navigation within complex spatial datasets using multiple interaction paradigms and is primarily intended for exploratory tasks such as spatial understanding and contextual inspection rather than analytical processing or data editing.

The system follows a small set of core principles. First, it is fully browser-based and relies exclusively on web technologies, allowing platform-independent access without local software installation. Second, it supports heterogeneous 3D data representations, including point clouds, meshes, and voxel-based models, which are handled within a unified scene and interaction framework. Scalability is addressed through spatial subdivision, multi-resolution representations, and demand-driven data loading, enabling interactive performance for large datasets under varying hardware conditions. Finally, the platform

emphasizes interaction-oriented exploration by integrating first-person navigation, immersive virtual reality, and alternative input modalities. Analytical processing and data modification are intentionally treated as external tasks, allowing GeoXperience to remain lightweight and focused on interactive exploration.

2.2 System Architecture

GeoXperience follows a client–server architecture designed. This separation of responsibilities enables efficient data management while allowing responsive interaction and visualization directly within the web browser.

The **server component** is responsible for data provisioning and metadata management. It provides access to preprocessed datasets that are organized into spatial chunks. Lightweight metadata describing spatial extents and available resolution levels is delivered to the client to support view-dependent data selection. Data transmission is performed on demand, ensuring that only those data segments required for the current viewpoint are transferred.

The **client component** runs entirely in the web browser and handles scene management, rendering, and user interaction. Using WebGL-based rendering, the client dynamically loads and unloads data chunks according to the current camera position, viewing direction, and interaction mode. The client maintains a local cache of recently used data to reduce redundant network requests and to improve interaction responsiveness.

Communication between client and server is based on standard web protocols. Metadata and data chunks are requested asynchronously using HTTP-based interfaces, allowing non-blocking data transmission during navigation. This asynchronous design ensures that rendering and interaction remain responsive even under varying network conditions.

Figure 1 illustrates the overall system architecture and data flow. After the selection of a particular dataset, the client first requests its metadata, which defines the spatial structure and available levels of detail. During interaction, view-dependent requests are issued to the server, which responds with the corresponding data chunks. The received data is then integrated into the scene graph and rendered according to the current performance budget.

This modular architecture allows the system to scale with dataset size and supports heterogeneous client devices. At the same time, it enables flexible extension of individual components, such as alternative data sources, rendering strategies, or interaction modules, without altering the overall system design.

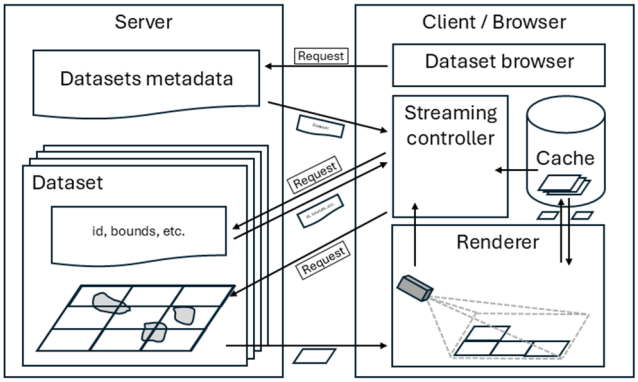

Figure 1 Overview of the client–server architecture of GeoXperience

2.3 Data Preparation and Organization

The presented approach relies on a lightweight but structured data preparation and organization workflow. The primary goal of this workflow is to reduce data transfer and support view-dependent loading without imposing extensive preprocessing requirements.

All supported input data, including point clouds, surface meshes, and voxel-based models, are organized using the same spatial abstraction. Independent of their representation, datasets are subdivided into spatially bounded units referred to as chunks. Each chunk represents a local subset of the dataset and serves as the basic unit for storage, transmission, and rendering. The chunking process is based on spatial subdivision of the dataset extent and does not depend on semantic or structural properties of the input data. As a result, the same chunking strategy is applied uniformly across all data types, enabling a consistent handling of heterogeneous 3D representations within a single scene.

For each chunk, lightweight metadata is generated to describe its spatial extent and resolution characteristics. This metadata includes bounding volumes and basic statistics such as element counts. The metadata is intentionally compact and is transmitted to the client after the dataset selection to describe the dataset structure and spatial layout.

Data preparation is designed to be independent of specific interaction modes, rendering techniques, or target devices. No assumptions are made regarding navigation paradigms or visualization styles at preprocessing time. Consequently, the same prepared dataset can be explored using different software, including desktop-based and virtual reality applications, without additional data transformation.

The separation of metadata and geometry data enables lazy loading, as chunks can be selected based on metadata without immediately transferring geometry. Based on the

current viewpoint and interaction context, the client requests only those chunks required for rendering, while the server remains responsible solely for delivering the requested data.

The organization of all supported data types into a unified chunk-based structure with associated metadata enables lazy loading, as chunks can be selected based on metadata without immediately transferring geometry. The loading strategy is described in the following sections.

2.4 Level-of-Detail and Streaming Strategy

Level-of-detail (LOD) selection and data streaming are performed dynamically on the client side and are tightly coupled to the chunk-based data organization described in the previous section. The overall goal of the strategy is to balance visual fidelity and rendering performance under varying viewing conditions while maintaining interactive frame rates across heterogeneous devices.

While all supported data types share the same spatial chunking and view-dependent selection framework, the representation and handling of LODs differ substantially between these data types.

For **point cloud data**, LODs are handled in an *additive* manner. Each chunk contains a shuffled set of points, allowing progressively larger subsets of the same chunk to be loaded and rendered. Lower LODs correspond to smaller point subsets, while higher LODs incrementally enrich the representation by adding more points. As a result, multiple LOD levels of the same chunk may coexist simultaneously in the scene, and visual refinement occurs gradually without replacing already rendered geometry.

In contrast, **mesh- and voxel-based datasets** rely on discrete geometry representations for each LOD level. For these data types, each chunk provides a set of mutually exclusive LOD variants, where higher levels represent more detailed geometry (e.g. increased triangle density or finer voxel resolution). At runtime, exactly one LOD representation per chunk is active at any given time. When the selected level of detail changes, the currently rendered representation is replaced by another LOD variant of the same chunk. This *replacement-based strategy* avoids redundant geometry and is better suited to the memory and rendering characteristics of surface- and volume-based representations.

Despite these differences, LOD selection follows a common conceptual process for all data types. First, candidate chunks are identified using view-dependent criteria such as camera frustum intersection and spatial proximity to the current camera position. This step limits further evaluation to those chunks that are potentially relevant for rendering. An example of the resulting spatial selection is shown in Figure 2, where the current camera position and the view frustum are visible.

In a second step, the desired level of detail is determined using a cost model. The estimated cost for a chunk representation is defined as a combination of its spatial distance to the camera and its selected resolution level. The distance term reflects the decreasing visual importance of distant geometry, while the resolution term accounts for the increasing rendering cost associated with higher LODs. Weighting factors allow the relative influence of distance and resolution to be adjusted, enabling different exploration styles such as prioritizing high detail in the immediate vicinity of the camera or achieving more uniform coverage of the visible scene. For a given chunk i, the estimated cost $c_i$is defined as

$$c_i = w_d \cdot d_i + w_{lod} \cdot c_{lod,i} \quad (1)$$

where d_i denotes the distance between the camera and the chunk, and w_d and w_{lod} control the relative influence of distance and resolution. The normalized level-of-detail term $c_{lod,i}$ is computed as

$$c_{lod,i} = \frac{lod_i}{lod_{max}} \quad (2)$$

with lod_i representing the selected resolution level and lod_{max} the maximum available level.

Figure 2 The currently loaded chunks shown from a top-down view. The colors encode the loaded LOD (orange: high, blue: low). The available but not loaded chunks are shown in black

For point clouds, this cost model governs the incremental enrichment of chunks by requesting additional point subsets

as long as sufficient rendering budget is available. For meshes and voxels, the same cost model determines which single LOD variant of a chunk should be active, and whether a chunk should be rendered at all. Chunks whose estimated cost exceeds the available budget may be temporarily omitted or rendered at a coarser resolution until performance conditions improve. The colored chunks in **Figure 2** illustrate this behavior.

The streaming of the chunks is then performed following a lazy loading principle, such that geometry data is requested from the server only when a chunk–LOD combination becomes relevant according to the current view-dependent selection. Already loaded data is cached and reused as long as it remains relevant, reducing redundant network transfers.

This strategy enables consistent, view-dependent level-of-detail management for heterogeneous 3D data and forms the basis for adaptive rendering and performance control described in the following section.

2.5 Adaptive Rendering and Performance Management

To ensure reasonable performance when using large-scale datasets mechanisms that adapt rendering effort to the capabilities of the client device and the current interaction context are required. In this approach, performance management is handled entirely on the client side and is closely coupled with the view-dependent streaming and LOD strategy described in the previous section.

The system maintains a global rendering budget that limits the overall rendering complexity at runtime. This budget constrains the amount of geometry that may be rendered simultaneously, for example in terms of the number of points, triangles, or volumetric elements. The budget is not fixed to a specific data type but applies uniformly across all chunk-based representations, ensuring consistent behavior for heterogeneous datasets.

Rendering performance is continuously monitored. Indicators such as frame rate stability and the current rendering load are used to assess whether the system operates within acceptable performance bounds. These bounds control the current available budget. If the frame rate drops below these bounds, the budget is decreased accordingly. If the estimated or observed rendering cost now exceeds the available budget, the client adapts its rendering strategy by reducing the selected LOD for affected chunks or by temporarily omitting more expensive chunks. Conversely, when sufficient performance is available, the budget is increased and more data may be requested and integrated. The adaptation is performed incrementally to avoid abrupt visual changes.

The adaptive rendering strategy is designed to accommodate heterogeneous client devices and interaction modes. Differences in available computational resources, graphics capabilities, and display characteristics are implicitly addressed through the budget-based selection process, without requiring device-specific configuration. As a result, the same dataset can be explored on desktop systems, mobile devices, or immersive VR setups while maintaining interactive performance.

2.6 Interaction and Navigation Concepts

Interaction and navigation are designed to support intuitive exploration of spatially complex environments while remaining independent of specific data representations.

Navigation is primarily realized through a first-person movement paradigm. The camera represents the user's viewpoint and can be translated and rotated freely within the environment. Movement is constrained by basic collision handling to prevent unintentional traversal through solid geometry or terrain surfaces. This navigation model allows users to explore scenes at human scale and facilitates intuitive understanding of spatial relationships. Figure 3 illustrates this perspective on a point cloud dataset.

Figure 3 A point cloud dataset from a first-person-view perspective. The points are colored by a combination of true and label colors

The system supports multiple input modalities that map onto the same underlying navigation and interaction concepts. Desktop-based interaction relies on mouse and keyboard input, while immersive exploration is enabled through virtual reality devices using WebXR. In addition, alternative input methods such as game controllers or gesture-based interaction can be integrated without altering the core navigation logic. This abstraction ensures consistent

interaction behavior across different devices and usage contexts.

Beyond navigation, the interaction model supports basic scene interaction tasks such as object selection and contextual information access. These interactions are designed to complement exploratory navigation and do not involve direct modification of the underlying data. By limiting the complexity of interactions, the system clearly focuses on exploration and avoids mixing navigation with analytical or editorial workflows.

3 Results and Experiments

3.1 Experimental Setup

The experimental evaluation aims to assess the performance and interactive behavior of GeoXperience under realistic exploration scenarios and across different device classes. The focus is placed on rendering performance, streaming behavior, and interaction responsiveness during navigation through large datasets.

Experiments were conducted using two representative datasets that differ in spatial extent, geometric complexity, and data representation. The datasets include point cloud and voxel data organized according to the chunk-based data structure described in Section 2.3.

To evaluate the impact of hardware heterogeneity, experiments were performed on different client devices, including a desktop workstation, a mobile or laptop-class system, and an immersive virtual reality setup. All experiments were executed in a standard web browser without local software installation. Network conditions correspond to typical broadband connectivity and were kept constant during each experiment run.

During the experiments, users navigated the datasets using first-person interaction and, where applicable, immersive virtual reality controls. Navigation scenarios included both continuous movement through the scene and stationary inspection of selected regions. These scenarios were chosen to reflect typical exploratory use cases and to trigger different streaming and level-of-detail transitions.

Performance measurements were recorded on the client side during interaction. Metrics include frame rate, the amount of geometry rendered, and data streaming activity over time. Measurements were collected continuously during navigation to capture dynamic behavior rather than isolated static viewpoints.

This experimental setup provides the basis for the quantitative and qualitative results presented in the following sections.

3.2 Test Datasets

The experimental evaluation was conducted using several datasets that represent typical application scenarios. The selected datasets differ in data representation, spatial extent, and geometric complexity, allowing the evaluation across heterogeneous input types while using the same chunk-based organization and streaming strategy.

Hannover-Linden-Nord

The first dataset consists of a large-scale lidar point cloud acquired by mobile mapping. The dataset covers a complete district of the city of Hanover, Germany and contains a high density of points, resulting in a substantial overall data volume of 136M points (see Figure 4). The point cloud is spatially subdivided into 765 chunks of uniform extent of 25 m. It further contains class label information at each point obtained from a classification. In this work, the labels are only used for visualization purposes.

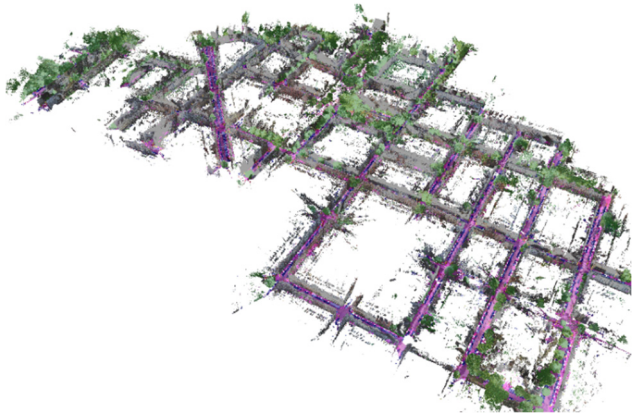

Figure 4 The point cloud dataset contains an entire district of the city of Hanover. The colors are a mixture of true colors and label colors (road, façade, vegetation, etc.)

Campus Dataset

In addition, a voxel-based dataset was included to evaluate volumetric representations (see Figure 5). The dataset was derived from a point cloud dataset, also collected via mobile mapping, and consists of approx. 9.6M voxels at the minimum LOD of 0.1 m voxel size. The data is organized into 349 spatial chunks that correspond to the same chunking scheme as the other datasets, enabling consistent view-dependent selection and streaming behavior. The voxel chunks provide multiple resolution levels that vary in voxel size.

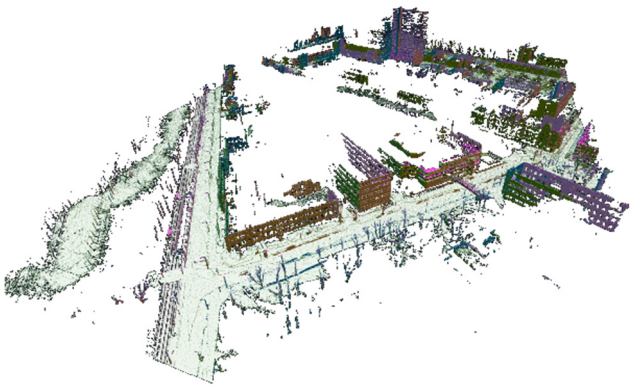

Figure 5 The voxel dataset shows a part of the campus of the Leibniz University of Hanover. The colors encode the normal directions per voxel

Both datasets were preprocessed in the same way. Metadata describing tile extents, available resolution levels, and basic complexity measures was generated during preprocessing. No dataset-specific optimizations or interaction-specific preprocessing steps were applied. This experimental setup allows an evaluation of the proposed methods under realistic and heterogeneous exploration conditions.

3.3 Performance Metrics

The performance evaluation is based on a set of client-side metrics that capture rendering efficiency, streaming behavior, and interaction responsiveness during exploration. The following metrics are recorded continuously during exploration scenarios to capture dynamic system behavior rather than isolated static viewpoints.

Frame rate (FPS) is used as the primary indicator of interactive performance and is recorded continuously during navigation. It reflects the combined impact of rendering load, streaming activity, and interaction handling.

Rendered geometry complexity is measured by the number of active elements (points, triangles, or voxels) rendered at runtime. This metric indicates how the adaptive level-of-detail strategy balances detail and rendering cost.

Streaming activity is characterized by the frequency of chunk and LOD requests as well as the amount of data transferred during navigation. This captures the behavior of the lazy loading and caching mechanisms.

Appearance is assessed qualitatively by inspecting the visible scene and its updates during navigation.

3.4 Performance Results

The performance results presented in this section are based on a comparison of different datasets evaluated on a single desktop system equipped with an Intel i5 CPU, 32 GB RAM, and an NVIDIA RTX 2080 GPU. This configuration serves as a reference platform to analyze dataset-dependent behavior while keeping hardware conditions constant.

The frame rate (FPS) over time for the evaluated datasets during interactive exploration is illustrated in Figure 6. Across both datasets, interactive frame rates are maintained throughout navigation. Variations in FPS correlate with changes in viewpoint and movement speed, reflecting view-dependent chunk selection and level-of-detail transitions. Short-term FPS drops are primarily associated with increased streaming activity when entering previously unseen regions and recover once the required data has been loaded. In the end, the target frame rate is maintained in both scenarios, although, especially when using the voxel dataset, the frame rate needs a certain time to adjust to the target frame rate.

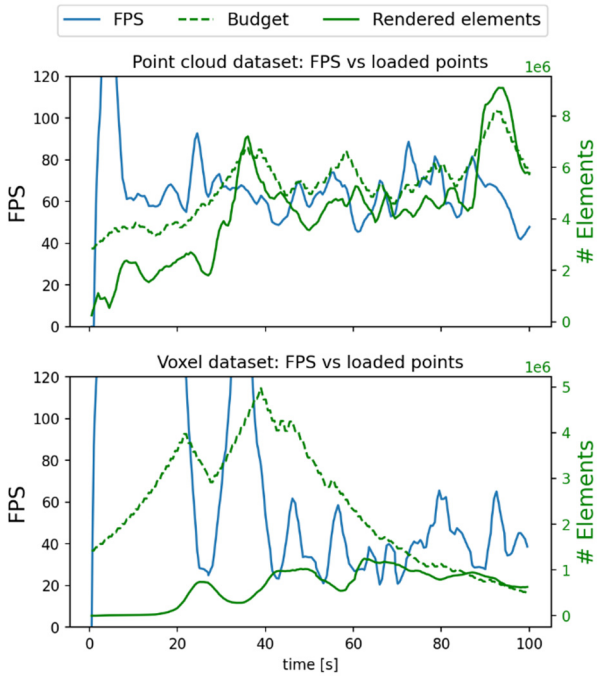

Figure 6 The development of frame rates (FPS) in relation to the budget and the number of points loaded while using the different datasets

In addition, in the first 30 seconds for the point cloud dataset and 60 seconds for the voxel dataset, the number of rendered points was significantly lower than the budget would have allowed. The reason for this is obviously a bottleneck in data transmission, as the scene is empty at the

beginning and all chunks must first be transferred from the server. Later on, there are only a few new chunks that are not cached and need to be transferred. From this point on, the budget and rendered points are very close to each other.

This is also confirmed by Figure 7, which depicts the **streaming activity** over time in terms of chunk and LOD requests and the data volume to be loaded. Streaming is triggered by camera movement, followed by periods of reduced activity as the view stabilizes. This pattern reflects the lazy loading strategy, where data is requested only when required for the current view, minimizing unnecessary data transfer.

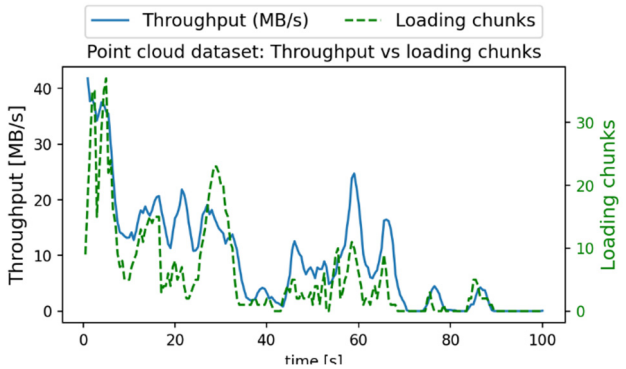

Figure 7 The streaming activity during the point cloud dataset. Due to caching of chunks, there is significantly less throughput after a certain number of chunks was loaded

In addition to quantitative performance metrics, visual appearance was inspected qualitatively for different datasets and levels of detail. Lower-resolution representations provide a coarse but coherent overview of the scene, while higher-resolution representations reveal fine geometric detail in regions close to the camera. Transitions between LODs occur progressively and do not result in abrupt visual changes. An example is shown in Figure 8.

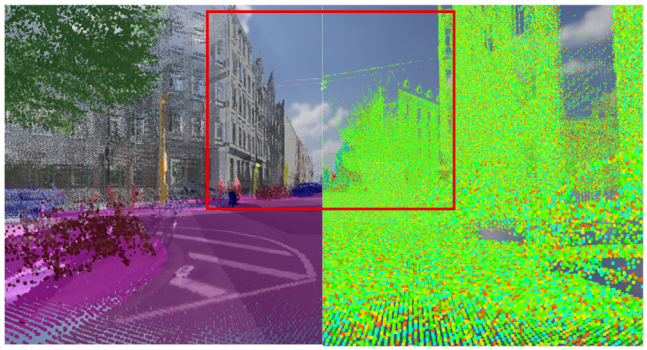

Figure 8 Left: a FPV on the Hannover-Linden-Nord dataset. Right: the same view showing the individual LODs in different colors. Although the chunks in the background (red marker) are rendered in lower LODs (some colors are missing), differences in point density are not noticeable

The influence of different weight settings in the cost function was also evaluated. Increasing the weight of the distance term prioritizes the loading of nearby chunks at higher resolution, resulting in detailed representations in the immediate vicinity of the camera while distant regions remain coarse. Conversely, emphasizing the LOD-related term leads to more uniform resolution across the visible scene at the expense of fine detail in close-range views. These results demonstrate that the cost function parameters provide intuitive control over the balance between local detail and global context.

3.5 Device Comparison

To assess the impact of hardware heterogeneity on interactive exploration, GeoXperience was evaluated across different classes of client devices. The comparison focuses on observable differences in rendering behavior and performance. As the point cloud dataset turned out to be the most challenging type, the data was collected during the first 100 seconds using the Hannover-Linden-Nord dataset. In this case, the geometry complexity is given by the number of points loaded and rendered. The target FPS was 60. Figure 9 shows the development of the frame rate regarding the budget and the number of actually rendered points.

On a **desktop system**, which was equipped with an i5 CPU, 32GB RAM and an RTX2070 GPU, the adaptive rendering strategy allows higher levels of geometric detail to be integrated. Higher LODs are reached more quickly, and a larger number of chunks can be maintained concurrently within the rendering budget. With maintaining the target of 60 fps, approx. 5M points were rendered.

On the **mobile** (Pixel 7, 8GB RAM) and **laptop device** (Intel Ultra 7 without dedicated GPU and 16GB RAM), interactive frame rates are preserved by operating at lower average levels of detail. Especially the mobile phone was obviously struggling more with the load compared to both other systems. The number of points to be rendered was significantly lower in both cases and was limited to approx. 1M (laptop) and approx. 0.4M (mobile phone).

In **VR** scenarios, performance constraints are stricter due to higher frame rate requirements and stereoscopic rendering. To prevent motion sickness the frame rate should not be lower than 90 FPS. In addition, the operation mode of the VR device has to be considered. If it is used independently, the rendering is done by the device itself. The performance of the device is comparable to a high-end mobile device. If the scene is rendered on a desktop computer and streamed to the VR device, the major load is already done on the computer. For the stand-alone VR

results shown in Figure 9, the target frame rate was set to 90 fps, which could be maintained after the starting period of approx. 30 seconds. The budget was then limited to 0.4M points.

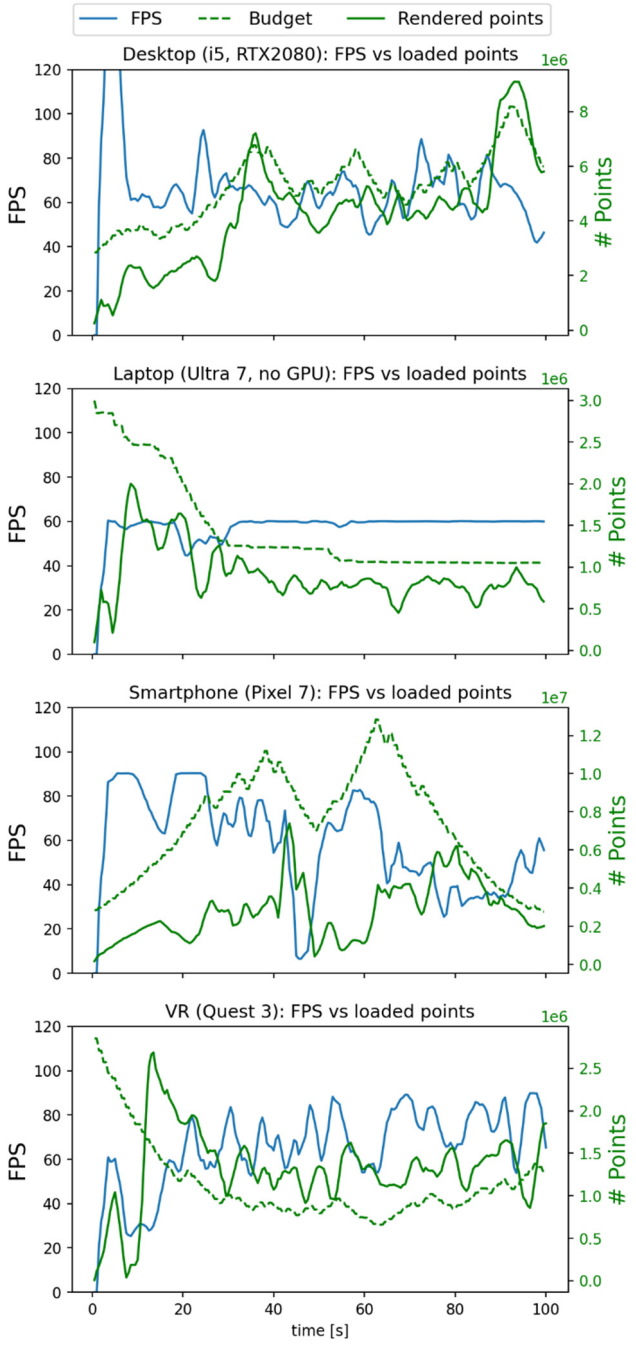

Figure 9 The development of the frame rate, the budget and the number of points as performance indicators collected during tests on different devices

3.6 Interaction Scenarios

To complement the quantitative performance evaluation, several interaction scenarios were examined to assess the practical behavior during exploratory use. The focus of these scenarios lies on navigation flow, responsiveness, and visual experience under different interaction modes rather than on numerical performance measures.

In a desktop-based scenario, users navigate the scene using mouse and keyboard or gamepad input in a first-person perspective. Continuous movement through the environment triggers view-dependent streaming and level-of-detail transitions. During this scenario, interaction remains responsive, and visual context is preserved through the immediate availability of lower-resolution representations. As the camera slows down or remains stationary, higher-resolution data is progressively integrated, enabling detailed inspection of local structures.

An immersive virtual reality scenario was used to evaluate interaction behavior under increased performance constraints and stereoscopic rendering. Navigation in VR emphasizes smooth and stable frame rates, which are maintained by prioritizing lower-resolution representations during movement. Higher-resolution content becomes visible primarily during stationary viewing phases, allowing users to inspect details without interrupting interaction.

An alternative input modality is gesture-based navigation. As shown in Figure 10, predefined gestures, like a raised arm for turning or closed or open hands for moving backwards and forwards, are mapped to the same underlying navigation and movement model. Switching between interaction modes does not alter the streaming or level-of-detail behavior, demonstrating that interaction handling is decoupled from data selection and rendering logic.

Figure 10 Gesture controls can be used to navigate in the scene

4 Discussion

The results presented in this work demonstrate that the proposed view-dependent streaming and adaptive rendering strategy enables interactive exploration of heterogeneous geospatial 3D datasets within a browser-based environment. At the same time, the experiments reveal several trade-offs and limitations that arise from the chosen design decisions and that should be considered when applying the approach to specific use cases.

Interactive frame rates are achieved primarily by prioritizing responsiveness over immediate visual completeness. Short-term frame rate drops occur when entering previously unseen regions and coincide with increased streaming activity. Although these effects are limited in duration and recover once the required data has been loaded, they highlight the inherent dependency of the approach on network latency and data availability. Lazy loading effectively reduces unnecessary data transfer, but it cannot entirely eliminate transient performance fluctuations under rapid or unpredictable navigation.

The adaptive level-of-detail strategy successfully regulates rendering complexity, but it also implies delayed visual refinement. During movement, lower-resolution representations dominate, while higher-resolution detail is progressively integrated during stationary inspection. This behavior supports smooth navigation and interaction continuity but may be less suitable for tasks that require immediate access to fine-scale detail across the entire visible scene. The results therefore reflect a deliberate trade-off between visual fidelity and interaction stability.

Differences in refinement behavior between point cloud, mesh, and voxel datasets further illustrate the influence of data representation. Progressive point cloud refinement leads to smooth but initially sparse visual appearances, whereas mesh- and voxel-based datasets exhibit more noticeable changes due to discrete replacement of chunk representations. While interaction remains uninterrupted in all cases, perceived visual stability varies depending on the underlying representation, indicating that the user experience is shaped not only by the streaming strategy but also by data characteristics.

The evaluation of different weight settings in the cost function highlights both the flexibility and sensitivity of the proposed approach. Emphasizing camera distance favors high-detail representations in the immediate vicinity of the viewer but can result in coarse background representations. Conversely, stronger weighting of the level-of-detail term leads to more uniform resolution across the scene at the expense of local detail. Although the parameters provide intuitive control over this balance, selecting suitable weights remains a non-trivial task and may require dataset- or application-specific tuning.

In comparison to related approaches, GeoXperience occupies a complementary position. Web-based frameworks such as Potree or CesiumJS provide efficient and well-established solutions for specific data types or spatial scales, often focusing on inspection-oriented or map-centric navigation concepts. Desktop-based tools remain indispensable for detailed analysis and data editing. In contrast, the approach presented here emphasizes immersive, experience-oriented exploration at local scale. Rather than replacing existing tools, GeoXperience extends the range of exploratory interaction scenarios.

The scope of the presented evaluation is subject to several limitations. Experiments were conducted on a limited set of representative datasets and controlled navigation scenarios. The results therefore do not provide guarantees for all possible dataset configurations, interaction patterns, or network conditions. Scenarios involving extremely dense data, highly dynamic movement, or constrained bandwidth may expose additional performance bottlenecks.

Overall, the discussion indicates that the proposed methods are well suited for interactive, exploratory scenarios that benefit from progressive refinement and immersive navigation. At the same time, the identified trade-offs and limitations highlight opportunities for future work.

5 Conclusion & Outlook

5.1 Conclusion

This paper presented GeoXperience, a web-based platform for interactive exploration of large-scale geospatial 3D data. Motivated by the increasing volume and complexity of contemporary 3D datasets, the platform was designed to support intuitive, immersive exploration directly within the browser.

The proposed approach combines adaptive data streaming, multi-resolution rendering, and flexible interaction concepts, enabling the exploration of heterogeneous data types such as point clouds, meshes, and voxel models across different devices. The experimental results demonstrate that the system supports real-time interaction under varying hardware conditions and provides multiple navigation paradigms tailored to different exploration scenarios.

By focusing on experience-oriented interaction rather than analytical processing, GeoXperience complements existing visualization frameworks and desktop-based tools, offering an accessible entry point for immersive geospatial exploration.

5.2 Outlook

Future work will focus on extending the platform toward collaborative exploration scenarios, including shared navigation, synchronized viewpoints, and user-generated annotations. Such functionality would support collaborative analysis, teaching, and public engagement scenarios, while also raising new challenges related to consistency, synchronization, and access control. In addition, systematic user studies are planned to evaluate the impact of immersive interaction paradigms on spatial understanding, navigation behavior, and task performance across different user groups and devices.

Further research will address the integration of dynamic and time-dependent datasets, enabling the exploration of evolving spatial phenomena such as urban change, environmental processes, or simulation results. Supporting temporal variation requires extensions to data management, streaming strategies, and interaction concepts in order to balance temporal coherence with interactive performance.

Finally, the combination of web-based visualization with augmented reality concepts represents a promising direction for bridging physical and virtual representations of geospatial environments. Integrating AR-based interaction could enable spatially situated exploration scenarios in which digital geospatial data is directly related to physical environments or tangible models. Together, these directions aim to further extend the platform toward more interactive, collaborative, and context-aware exploration of geospatial 3D data.

References

Buyukdemircioglu, M., & Kocaman, S. (2018). A 3D Campus Application Based on City Models and WebGL. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-5, 161–165. https://doi.org/10.5194/isprs-archives-XLII-5-161-2018

Cabello, R., et al. (2025). Three.js – JavaScript 3D Library. https://threejs.org

CesiumGS (2024). CesiumJS – An open-source JavaScript library for 3D geospatial visualization. https://cesium.com/platform/cesiumjs/

Cignoni, P., Muntoni, A., Ranzuglia, G., & Callieri, M. (2021). MeshLab. Zenodo. https://doi.org/10.5281/zenodo.5114037

Esri (2024). ArcGIS Pro – 3D and Point Cloud Visualization

Giro3D Development Team (2024). Giro3D: A WebGL-based framework for geospatial 3D visualization. https://giro3d.org

La Guardia, M., Koeva, M., D'Ippolito, F., & Karam, S. (2022). 3D Data Integration for Web Based Open Source WebGL Interactive Visualisation. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVIII-4/W4, 89–94. https://doi.org/10.5194/isprs-archives-XLVIII-4-W4-2022-89-2022

Lee, A., & Jang, I. (2019). Implementation of an open platform for 3D spatial information based on WebGL. ETRI Journal, 41(3), 277–288. https://doi.org/10.4218/etrij.2018-0352

OpenGeo (2024). maplibre-gl-lidar: LiDAR point cloud visualization for MapLibre GL JS. https://opengeos.org/maplibre-gl-lidar/

QGIS Development Team (2024). QGIS Geographic Information System.

Schütz, M., Ohrhallinger, S., & Wimmer, M. (2020). Fast Out-of-Core Octree Generation for Massive Point Clouds. Computer Graphics Forum, 39(7), 155–167. https://doi.org/10.1111/cgf.14134