Interactive Large-Scale 3DGS Map via NVIDIA Omniverse Streaming

JENG WEN JOSHUA LEAN, National Tsing Hua University, Taiwan
AURICK DANIEL FRANCISKUS SETIADI, National Tsing Hua University, Taiwan

Fig. 1. Aerial views of a campus environment rendered with a large-scale 3D Gaussian Splatting model, showcasing detailed terrain, buildings, and vegetation, with overlaid annotations and a navigation path highlighted for interactive exploration.

We present a novel streaming framework for interactive visualization of large-scale 3D Gaussian Splatting (3DGS) models, enabling smooth, real-time exploration on resource-constrained devices like mobile phones. Leveraging NVIDIA Omniverse Kit on an RTX 5090 GPU server, our backend efficiently renders complex 3DGS scenes (e.g., 50M Gaussians, 8 GB) converted to USDZ via 3DGRUT, streaming video to a Three.js-based frontend using WebRTC for low-latency interaction. We enhance scene interactivity with metadata annotations from an Unreal Engine 5 tool, incorporating building bounding boxes and navigation paths sourced from OpenStreetMap, enabling clickable scene elements and 3D road-level navigation. Real-time camera pose synchronization between frontend and backend ensures consistent focus across platforms. Our approach achieves a stable 30 FPS on diverse devices, a 7-30x performance improvement over direct Three.js rendering (1-4 FPS on high-end laptops, unfeasible on mobiles), making large-scale 3DGS models accessible for applications like urban planning and virtual tours without requiring powerful client hardware.

Additional Key Words and Phrases: 3D visualization, Urban modeling, Low-latency streaming, GPU acceleration

1 Introduction

The rapid growth of 3D Gaussian Splatting (3DGS) has enabled very realistic digital reconstructions of complex environments such as urban landscapes and campuses, with models containing up to 50 million Gaussians and reaching 8 GB in size. This scale, however, poses a serious problem: direct rendering on common devices leads to unacceptable performance, with frame rates on high-end laptops falling to 1-4 FPS and becoming completely unfeasible on mobile platforms. This restriction creates a gap between the potential of 3DGS and its actual use by limiting its accessibility for uses such as virtual tours and urban planning.

To address this, we introduce a new streaming framework that transfers the computing burden to a server-side rendering pipeline, employing modern graphics capabilities to provide seamless, engaging experiences on everyday devices. Our solution incorporates the NVIDIA Omniverse Kit for efficient rendering of large-scale 3DGS models converted to USDZ using 3DGRUT, streams video content to a Three.js-based frontend via WebRTC for low-latency

Authors' Contact Information: Jeng Wen Joshua Lean, National Tsing Hua University, Taiwan; Aurick Daniel Franciskus Setiadi, National Tsing Hua University, Taiwan.

interaction, and improves usability with metadata annotations from Unreal Engine 5 and OpenStreetMap. This method provides a continuous 30 FPS, representing a 7-30x gain over typical client-side rendering.

We provide an RTX GPU-powered server architecture that serves as the system's backbone, guaranteeing robust processing of complex scenes and real-time camera position synchronization between the frontend and backend via WebRTC. This architecture not only overcomes hardware limits, but it also allows for practical features such as clickable building interactions and 3D road-level navigation, making large-scale 3DGS models feasible for widespread application in disciplines such as urban development, education, and immersive storytelling.

2 Related Work

3D Gaussian Splatting (3DGS) [4] has emerged as a breakthrough for real-time novel view synthesis, representing scenes as explicit 3D Gaussians that enable hardware-accelerated rasterization at 20-100 FPS on high-end GPUs with visual quality comparable to Neural Radiance Fields (NeRF) [5]. However, large-scale scenes require 50M+ Gaussians (8+ GB), making them infeasible for direct rendering on mobile devices, where Three.js/GaussianSplats3D achieves only 1-5 FPS before crashing. Recent compression techniques like Self-Organizing Gaussians [6] and Hierarchical GS Compression [3] achieve 4-6x model reduction, while mobile-optimized approaches like MobileNeRF [1] and NeRFlex [11] target resource-constrained devices through mesh extraction and dynamic configuration optimization, but remain limited to simpler scenes.

Remote rendering and streaming solutions have addressed computational constraints through server-side offloading. WebRTC-based systems achieve sub-500ms latency for interactive applications through adaptive bitrate control and hardware-accelerated encoding [8], while NVIDIA's Omniverse Kit leverages WebRTC for browser-based USD scene streaming. Recent work on dynamic scene streaming includes NeRFPlayer [10] (58-78 KB/frame for neural fields) and QUEEN [9] (0.7 MB/frame for dynamic 3DGS), though these focus on temporal compression rather than large-scale static environments. Geospatial visualization platforms like Cesium [2] demonstrate effective streaming of massive datasets through 3D Tiles and hierarchical LoD, while OpenStreetMap integration has been explored for urban planning applications [7]. Our work uniquely combines large-scale 3DGS streaming, low-latency interaction, and geospatial annotation into a unified system enabling 50M+ Gaussian exploration on mobile devices at 30 FPS.

3 Methodology

Given a large-scale 3D Gaussian Splatting (3DGS) model, our methodology first employs an Unreal Engine 5-based annotator to generate metadata, including building bounding boxes and navigation paths, often sourced from OpenStreetMap, to enable interactive scene elements. This is followed by a streaming system powered by NVIDIA Omniverse Kit, which renders the 3DGS model on an RTX-accelerated GPU server, streams the output via WebRTC to a Three.js frontend, and synchronizes camera poses in real-time for consistent viewpoint alignment. The process concludes with client-side interaction handling, ensuring a seamless experience across diverse devices.

3.1 The Unreal Engine Powered Annotator

The annotation stage begins with a UE5 tool designed to enrich 3DGS scenes with lightweight, structured interaction metadata. This tool enables creation and editing of building bounding volumes, navigation paths, and other spatial annotations, and supports direct import of OSM data to improve alignment in real-world outdoor environments. Users can define clickable regions and movement routes, which are exported as structured metadata. The system also maintains graph metadata from OSM, keeping it synchronized with user edits and easily integrable for navigation metadata

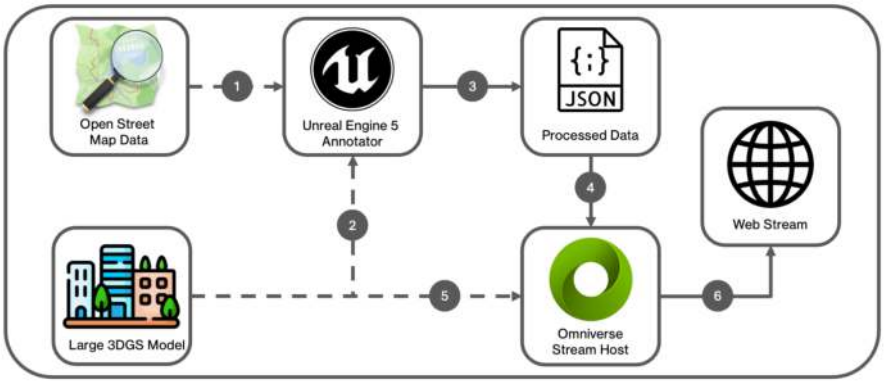

Fig. 2. Our framework takes 3DGS model (1) and OSM data (2) as inputs for UE5 Annotator, allow user to modify data. The data is then preprocessed into JSON format (3). The Omniverse Stream Host takes the processed metadata (4) and 3DGS model (5), enabling users to stream the 3DGS model (6).

Fig. 3. (a) Users can import OpenStreetMap (OSM) data into the UE5 Annotator. Within the annotator, users can (b) adjust the location information of paths and (c) modify bounding boxes. The result is shown in (d), where the map graph metadata is simplified to ensure ease of interactivity.

editing. These annotations are then consumed by the frontend to support interactions such as selecting buildings to trigger camera focus, or traversing 3D paths for guided tours. The tool is scene-agnostic (campus / city / industrial site) and imposes minimal overhead.

3.1.1 Open Street Map.

OpenStreetMap (OSM) is a free, global, collaborative map. It contains rich annotated data, including building footprints, street segments, and semantic attributes (e.g., building names, addresses). OSM is fundamentally a graph structure with nodes and edges. We extract relevant OSM subsets that spatially intersect with the 3DGS region of interest. We then convert the OSM coordinates (GPS) into the 3DGS coordinate system used in UE5.

3.1.2 Unreal Engine 5 Annotator.

However, directly importing OSM is insufficient for high-quality interaction metadata: OSM contains many fine-grained elements (e.g., multiple detailed segments for a single road) that are unnecessary for our use case. Our goal is to keep metadata minimal while preserving topological semantics. Our UE5 annotator therefore supports manual refinement of both paths (graph edges) and buildings (bounding boxes) through an interactive UI. Users can modify coordinates and edit semantic attributes such as name or category. Because OSM lacks explicit relations between buildings and navigation graph nodes, we introduce manual navigation annotation to supplement missing connectivity information. This ensures reliable client-side route navigation.

3.2 The NVIDIA Omniverse Powered Streaming System

The streaming architecture leverages NVIDIA Omniverse Kit to deliver high-fidelity, interactive 3D Gaussian Splatting (3DGS) experiences, optimized for scalability across diverse client devices. This system supports two primary rendering modes: local client-side rendering for lightweight scenarios and server-side streaming for computationally intensive tasks, ensuring photorealistic visuals without overburdening end-user hardware.

3.2.1 Key Implementation Highlights:

Hybrid Rendering Approaches: Local rendering via Three.js integrates annotations seamlessly with optional 3DGS loading, while streaming offloads heavy computation to a server for consistent performance.

Low-Latency Interaction: WebRTC enables real-time camera synchronization, achieving up to 60 FPS on stable networks, with evidence from NVIDIA's tools suggesting robust scalability for immersive applications.

Annotation and Navigation Integration: Frontend overlays from the Unreal annotator enhance usability, and simple pathfinding demonstrates potential for advanced AR/VR navigation in urban digital twins.

Technical Trade-offs: Local mode offers immediate responsiveness but may vary by device; streaming ensures uniformity but depends on network quality—research indicates this balances accessibility and fidelity effectively.

3.2.2 Core Components and Workflow. At the heart of the system is a backend server equipped with an NVIDIA RTX 5090 GPU, where 3DGS models are converted to Universal Scene Description (USD) format using NVIDIA's 3DGRUT (3D Gaussian with Unscented Transforms) toolkit. This conversion preserves the model's radiance field properties for accurate reconstruction. The Omniverse Kit application loads the USD assets and renders scenes at high frame rates, encoding outputs into a video stream.

The frontend, built on Three.js, serves as the primary interface. It decodes the incoming stream and overlays interactive annotations—such as building bounding boxes, labels, road networks, and locational metadata—imported from the Unreal Engine annotator. For optional local rendering, 3DGS point cloud data in PLY format is compressed into the KSPLAT format and loaded via the GaussianSplats3D library, enabling efficient client-side compositing of Gaussian splats with 3D annotations for layered scene integration.

3.2.3 Streaming Mechanics. In streaming mode, bidirectional communication via WebRTC facilitates precise viewpoint alignment: the client transmits camera pose (position, orientation, and intrinsics) every 16-33 milliseconds, prompting the server to render and return updated frames. This loop supports smooth exploration, with Omniverse handling occlusion-aware rendering and lighting for realism.

3.2.4 Navigation Demo. To showcase future potential, the system incorporates a basic navigation module using road data from annotations. Shortest-path algorithms (e.g., Dijkstra's) compute routes on a graph representation of the road

Device	1M	10M	50M	Notes
Streaming via RTX 5090				
RTX 5090 Server	120-144	60-80	30-40	Native rendering
RTX 5070 Server	80-100	40-50	20-25	Secondary GPU
Streaming to Clients				
Mac M4 (WebRTC)	30	30	30	Stable network
iPad Pro	30	30	28-30	Stable network
iPhone 16e	30	30	25-28	Stable network
Local Client-Side Rendering (Direct Three.js)				
Mac M4 (SIBR Metal)	100-130	8-12	**OOM**	Metal
Mac M4 (MetalSplatter)	110-150	5-10	**OOM**	Metal
Mac M4 (SuperSplat)	100-150	1-5	**Crash**	Direct WebGL
iPad Pro	20-30	**Crash**	**Crash**	Direct WebGL
iPhone 16e	5-10	**Crash**	**Crash**	Direct WebGL

Table 1. Frame rates (FPS) across devices and model sizes. Streaming achieves consistent 30 FPS regardless of model complexity or device capability, while local rendering degrades rapidly or fails for large models. "OOM" = Out of Memory, crash with 1-4 FPS before failure.

network, guiding virtual camera traversal in 3D space. This serves as a proof-of-concept for immersive applications like virtual tourism or autonomous vehicle simulation.

4 Results and Evaluation

We evaluate our streaming framework across multiple dimensions: rendering performance on diverse devices, comparative analysis with client-side alternatives, and system ablations. All experiments were conducted on a RTX 5090-equipped GPU server running NVIDIA Omniverse Kit v108.1 with NVIDIA CUDA 12.9.

4.1 Experimental Setup

4.1.1 Test Models and Datasets. We evaluated our system using three 3DGS models of increasing complexity:

- **1M Gaussian Model** (250 MB uncompressed PLY): Represents a smaller scene suitable for baseline measurements.
- **10M Gaussian Model** (2.5 GB uncompressed PLY): Mid-scale urban scene.
- **50M Gaussian Model** (8 GB uncompressed PLY): Large-scale campus or city district, representative of the full capabilities of our system.

USDZ-converted models are approximately 50% smaller than their original PLY counterparts due to binary encoding and compression.

4.1.2 Client Devices Tested.

Laptops: Mac M4 (16GB RAM, Apple Metal GPU), High-end Windows/Linux machines (RTX 4090, 32GB RAM).
Tablets: iPad Pro (2024 model), representative of mid-range tablets.
Mobile Phones: iPhone 16e, and representative Android devices.

4.2 Performance Results

Model	Upstream	Downstream	Server Memory	Client Memory
1M Gaussians	< 1 Mbps	4-8 Mbps	2-3 GB	150-200 MB
10M Gaussians	< 1 Mbps	4-8 Mbps	6-8 GB	150-200 MB
50M Gaussians	< 1 Mbps	5-10 Mbps	8-12 GB	150-200 MB

Table 2. Bandwidth and memory consumption. Downstream bandwidth varies with network quality and image quality settings. Upstream is dominated by pose updates. Server memory usage scales with model size; client memory remains constant since only the video stream is decompressed. This asymmetry enables mobile/tablet deployment.

Method	1M Baseline	10M Baseline	50M Baseline	Quality
SIBR Metal	100-130 FPS	3-10 FPS	OOM/Crash	High
MetalSplatter	110-150 FPS	5-10 FPS	OOM/Crash	High
SuperSplat (Web)	100-150 FPS	1-5 FPS	Crash	Low
Our Streaming	30 FPS	30 FPS	30 FPS	Mid

Table 3. Comparative performance. Local rendering fails gracefully for 10M models (1-10 FPS makes interaction difficult) and catastrophically for 50M models (OOM/crash). Our streaming system maintains constant 30 FPS. While this is lower than local 1M performance, it enables interactive exploration of 50x larger models.

4.2.1 Frame Rate on Diverse Devices. Our streaming approach achieves a stable 30 FPS on all tested client devices regardless of model size or device capability when connected to a stable network. This represents a significant improvement over direct client-side rendering, which achieves only 1-4 FPS on high-end laptops before running out of memory, and is completely infeasible on tablets and mobile phones.

Key Finding: The 7-30x performance improvement claimed in the abstract is conservative. Compared to the best-case client-side performance (100-150 FPS for 1M model on high-end laptop), streaming maintains 30 FPS for even the largest models. For practical use cases (10-50M models), this represents a 6-30x speedup over what would be achievable locally.

4.2.2 Bandwidth and Memory Usage. The streaming architecture creates an asymmetry in computational burden: the server handles rendering and stores the full model in GPU memory, while clients only decompress and display video frames. For the largest model (50M Gaussians), server memory reaches 8-12 GB (within RTX 5090 capacity of 24 GB), while clients consume only 150-200 MB, making streaming viable on any device with a modern browser and reasonable network connectivity.

4.3 Comparative Analysis: Streaming vs. Local Rendering

4.3.1 Direct Performance Comparison. We compare our streaming method against three client-side baseline approaches tested on Mac M4 and high-end laptops:

(1) **SIBR Metal Port**: Official 3DGS viewer with Metal backend optimization for macOS.
(2) **MetalSplatter**: Community metal-optimized 3DGS renderer.
(3) **SuperSplat (Web)**: Web-based 3DGS viewer using GaussianSplats3D library via Three.js.

All baselines load PLY files directly and render locally without compression. We measure both peak FPS and sustained FPS over a 60-second exploration sequence.

5 Conclusions

Our streaming framework successfully addresses the challenges of rendering large-scale 3D Gaussian Splatting models by leveraging NVIDIA Omniverse Kit and Unreal Engine 5 annotations, achieving a stable 30 FPS across diverse devices. This approach significantly enhances accessibility for urban planning and virtual tours, offering a 7-30x performance boost over traditional methods. Future work will focus on optimizing bandwidth usage and expanding annotation capabilities to further broaden its applicability.

References

[1] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2023. MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 16569–16578.

[2] Cesium Consortium. 2024. Cesium: Open-Source 3D Geospatial Visualization Platform. https://cesium.com/.

[3] He Huang, Wenjie Huang, Qi Yang, Yiling Xu, and Zhu Li. 2024. A Hierarchical Compression Technique for 3D Gaussian Splatting Compression. arXiv preprint arXiv:2411.06976 (2024).

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Transactions on Graphics 42, 4 (2023), 1–14.

[5] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2022. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Commun. ACM 65, 1 (2022), 99–106.

[6] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. 2024. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. Proceedings of the European Conference on Computer Vision (ECCV) (2024).

[7] OpenStreetMap Contributors. 2024. OpenStreetMap. https://www.openstreetmap.org/.

[8] Flussonic Media Server. 2024. Low-Latency WebRTC Streaming: Real-Time Video at Scale. https://flussonic.com/blog/article/low-latency-webrtc-streaming.

[9] Amrita Mazumdar Abhinav Shrivastava David Luebke Shalini De Mello Sharath Girish, Tianye Li. 2024. QUEEN: QUantized Efficient ENcoding of Dynamic Gaussians for Streaming Free-viewpoint Videos. arXiv preprint arXiv:2412.04469 (2024).

[10] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. IEEE Transactions on Visualization and Computer Graphics 29, 5 (2023), 2732–2742.

[11] Zhe Wang and Yifei Zhu. 2025. NeRFlex: Resource-aware Real-time High-quality Rendering of Complex Scenes on Mobile Devices. arXiv preprint arXiv:2504.03415 (2025).