SceneHub4D: A Dataset and Evaluation Framework for 6-DoF 4D VR Scenes

Jaehong Kim, Tao Jin, Mallesham Dasari, Srinivasan Seshan, and Anthony Rowe

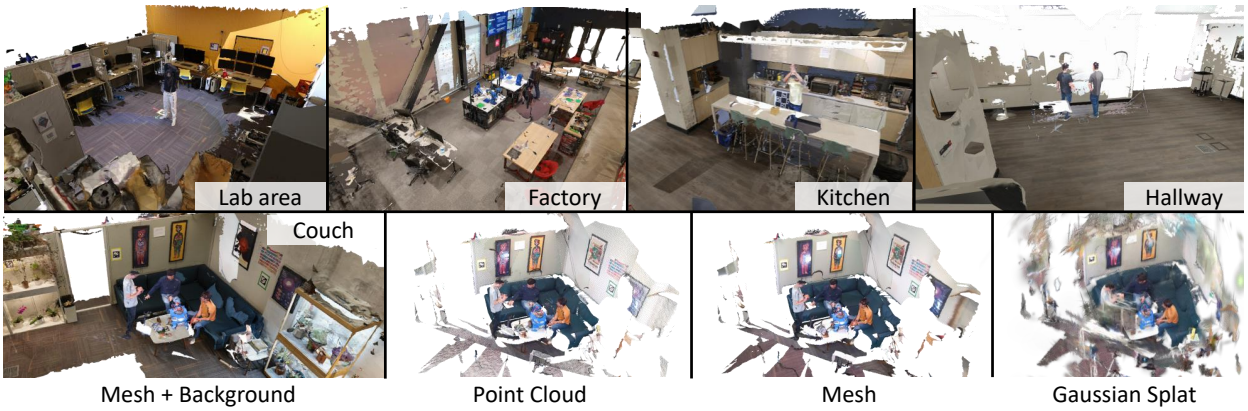


Fig. 1: Sample frames of the released dataset sequence. Project page: https://scenehub4d.github.io/.

Abstract—Volumetric video and 6-DoF scene capture are becoming central to immersive applications such as telepresence and mixed reality content delivery. However, existing volumetric datasets are often short in duration, restricted to studio-captured human subjects, and provide only limited geometric representations. Consequently, evaluating real-world immersive applications in full-scene contexts often necessitates custom capture and 3D reconstruction setups, creating high practical barriers and ultimately hindering reproducibility. To this end, we present SceneHub4D, a new dataset and evaluation framework. Our dataset captures long, dynamic sequences across diverse real-world indoor environments with synchronized multi-view RGB-D streams, calibrated camera poses, and high-resolution background geometry reconstructed via photogrammetry and LiDAR. We provide multiple 3D representations, including point clouds, textured meshes, and Gaussian splats, along with a software toolkit for format conversion, rendering, and metric evaluation. To support structured comparison and perceptual analysis, we provide supplementary metrics including Geometry Complexity Score and Volumetric Temporal Information, and evaluate rendering performance across desktop GPUs and VR headsets.
By lowering the practical barriers to capture, reconstruction, and evaluation, SceneHub4D enables researchers to study immersive 3D streaming and rendering systems without requiring custom hardware setups or complex data collection pipelines. We expect it will serve as a useful foundation for advancing volumetric media research.

Index Terms—6-DoF, 4D, Volumetric Video, Dataset, Toolkits, Metrics

1 Introduction

Recent advances in 3D sensing, rendering, and head-mounted displays have accelerated research in volumetric video, enabling applications such as telepresence, remote collaboration, and mixed reality content creation. As volumetric media systems move toward real-time performance and view-dependent rendering, the need for high-quality, reusable datasets for benchmarking has become increasingly urgent.

Despite progress in capturing and reconstructing 3D scenes, existing volumetric video datasets fall short in several key aspects. Many are limited to object-centric recordings in controlled studio environments (e.g., human objects, dance performances, etc), lacking the spatial complexity and background context required for evaluating real-world applications [20, 25, 33, 36]. In addition, they typically provide only one preprocessed geometry format, often a point cloud, without access to the original RGB-D data or camera poses, which hinders reconstruction, analysis, or conversion to other modern representations such as Gaussian splats [25, 33].

Moreover, these datasets do not offer metrics to describe, analyze, or compare the structural or perceptual characteristics of volumetric content. Descriptions are often qualitative, and sequence durations are short, typically a few seconds, limiting their usefulness for evaluating volumetric streaming systems that require longer, coherent sequences.

To address these gaps, we present SceneHub4D, a new volumetric video dataset and benchmarking framework for 6-DoF 4D scenes. SceneHub4D provides the following key contributions. First, it offers diverse raw RGB-D sequences with calibrated camera poses across indoor scenes lasting over 60 seconds. These captures include activities such as group conversations, object interactions, and movement in realistic environments (labs, kitchens, hallways, living rooms). To complement the sparse depth maps, we additionally provide a high-resolution, photogrammetry-based textured mesh of the entire scene. The mesh is globally aligned to the RGB-D coordinate system, enabling its use as a backdrop for rendering and ensuring scene completeness.

- *Jaehong Kim is with the Department of Artificial Intelligence, Inha University. This work was initiated while the author was with Carnegie Mellon University. E-mail: jaehong.kim@inha.ac.kr.*
- *Tao Jin is with Electrical and Computer Engineering, Carnegie Mellon University. E-mail: taojin@andrew.cmu.edu.*
- *Mallesham Dasari is with Electrical and Computer Engineering, Northeastern University. E-mail: m.dasari@northeastern.edu.*
- *Srinivasan Seshan is with Computer Science Department, Carnegie Mellon University. E-mail: srini@cs.cmu.edu.*
- *Anthony Rowe is with Electrical and Computer Engineering, Carnegie Mellon University and Bosch Research. E-mail: agr@andrew.cmu.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Dataset	Camera Pose	Per-view Depth	Point Cloud	Mesh	Gaussian Splat	Multi-person	Interactive Objects	Full Scenes	Photogramm. Background	Video Length	Metric	Software Suite
SceneHub4D	✓	✓	✓	✓	✓	✓	✓	✓	✓	≈2 min	✓	✓
Panoptic [20]	✓	✓	✓	-	✓‡	✓	✓	✓*	-	≈6 min	✓	✓
UVG-CWI-DQPC [11]	✓	✓	✓	✓	-	✓	✓	-	-	≈7 sec	-	✓
PCVD [25]	-	-	✓	✓	-	✓	✓	-	-	≈25 sec	✓	-
FSVVD [16]†	-	✓	✓	-	-	✓	✓	✓	-	≈13 sec	-	-
CWIPC [33]	-	✓	✓	-	-	✓	✓	-	-	≈45 sec	-	✓

Table 1: Comparison of volumetric video datasets. ‡: Later adopted in a dynamic Gaussian splatting study [29]. *: Provides capture in a dome environment. †: Dataset not publicly accessible at the time of this writing. Video length is estimated assuming 30 fps from the reported frame counts.

SceneHub4D supports conversion to multiple geometry representations.

Second, SceneHub4D provides a set of quantitative descriptors to facilitate structured comparison of dataset content. Specifically, we characterize geometric complexity using a Geometry Complexity Score (GCS), extend it with SSIM-aware GCS to incorporate perceptual quality across multiple views, and measure temporal variation using Volumetric Temporal Information (V-TI) based on point-wise displacements between aligned point clouds. With the dataset and proposed metrics, SceneHub4D enables compression benchmarking using raw formats compatible with geometry and video codecs, without requiring complex multi-camera calibration or synchronization pipelines.

Finally, SceneHub4D bridges directly to immersive VR applications. We provide a Unity-based application for live-like replay and validate our metrics by measuring rendering performance across desktop GPUs and standalone VR headsets (Quest 2, Quest 3), highlighting trade-offs between geometric complexity, perceptual quality, and real-time frame rates.

All together, SceneHub4D provides a comprehensive foundation for benchmarking immersive 3D content and advancing end-to-end VR system design. In the remainder of this paper, we describe the design of SceneHub4D, including data capture, geometry processing, and camera pose generation (§ 3). Next, we present a detailed dataset description and analysis with proposed evaluation metrics (§ 4). We further demonstrate representative use cases such as compression and streaming benchmarks (§ 5).

2 Related Works

A variety of volumetric video (also known as *free-viewpoint video*) datasets have been proposed, each highlighting different aspects of 3D capture such as interaction types, scene complexity, and representation formats. Tab. 1 summarizes the differences between our dataset and existing datasets.

Recent point cloud datasets such as **PCVD** [25], **CWIPC-SXR** [33], and **UVG-CWI-DQPC** [11] focus primarily on human captures using multi-view RGB-D sensors. They provide dense point cloud captures of social interactions using multi-camera RGB-D setups. However, PCVD and CWIPC-SXR do not provide explicit per-view camera pose annotations, making it difficult to reproject or evaluate flexible 3D representations such as meshes or emerging formats like Gaussian splats. While UVG-CWI-DQPC includes calibration and transformation metadata, all three datasets lack explicit background geometry, which is often necessary for evaluating full-scene rendering and real-world application contexts [19, 36].

Panoptic Studio [20] offers one of the earliest large-scale multi-person volumetric video datasets, captured using 10 Kinect v2 sensors inside a geodesic dome. While it enables rich social interaction capture, the confined dome environment and the use of older-generation sensors constrain geometric fidelity and spatial variety. Additionally, it lacks mesh or photogrammetric reconstructions.

FSVVD [16] is the dataset most comparable to ours, as it captures human–object interactions in diverse indoor environments with full-scene context. While a download link is provided on the project webpage [9], the dataset was not readily accessible at the time of our experiments due to platform-dependent access restrictions, which limits practical usability. In contrast, our dataset is hosted on an Open Storage Network (OSN) platform backed by S3 storage, with a partial subset additionally mirrored via a public Dropbox link, and is directly accessible from multiple regions. We further verified that the download bandwidth is stable and sufficient for practical use, enabling reproducible access across continents (Tab. 3).

Moreover, FSVVD does not provide mesh or photogrammetric geometry, and only reconstructed point clouds are available. Camera calibration parameters are not included, which restricts reuse in reconstruction or re-rendering pipelines. Similar to our dataset, FSVVD is captured using RGB-D cameras, which inherently introduces depth holes and sparsity. However, our dataset additionally provides photogrammetry-based geometry to compensate for such missing regions. Finally, FSVVD sequences remain relatively short in duration (≈13 sec), whereas our dataset focuses on longer continuous captures.

SceneHub4D complements prior efforts by providing a comprehensive collection of 3D representations, captured across diverse, real-world indoor environments. Each sequence includes per-camera depth, calibrated camera poses, and full-scene geometry with background. We also provide evaluation metrics and a software toolkit for flexible representation.

3 Dataset Generation

3.1 Data Capture

We describe the hardware and software setup used for synchronized multi-view RGB-D data capture, including calibration, synchronization, and photogrammetric background capture.

Capture setup. We use four Azure Kinect cameras to capture high-resolution RGB-D data at 30 fps. Each device records 1920×1080 RGB images and 640×576 depth maps, which are aligned to the RGB resolution using the Azure Kinect SDK [1]. Each scene sequence is captured over approximately 2 minutes on average. The cameras are placed strategically around the room—facing inward and spaced according to room constraints—to enable volumetric coverage. The capture environment dimensions are summarized in Tab. 4, and the camera placement from one of the scenes (lab area) is visualized in Fig. 2. To improve texture quality and reduce lighting variation, we attach lightweight diffusion LED light panels to the tripod of each camera.

Photogrammetry capture. For each scene, we captured approximately 1,000 high-resolution photographs using a fixed camera rig composed of five Ricoh cameras. To enhance geometric accuracy, we also performed a LiDAR scan using a Leica BLK360 laser scanner, generating a dense point cloud of the scene.

The captured images and LiDAR data were processed using RealityCapture [4], a photogrammetry software that reconstructs 3D geometry by aligning images via feature matching, estimating camera poses, and generating a textured mesh. By integrating visual data and laser scans, the software enables accurate and high-quality 3D reconstructions, which serve as background assets in our dataset.

Calibration. To ensure high-quality 3D reconstruction from multi-camera RGB-D data, accurate calibration is essential for both the intrinsic parameters of each camera and the extrinsic transformations between them. For the camera intrinsic calibration, we obtain each camera's intrinsics parameters, such as focal length, principal point,

and lens distortion, employing Zhang's method [44] from a set of flat checkboard pattern images. In addition, we adopt the Brown-Conrady distortion model [8] to account for radial and tangential lens distortion, which better reflects the optical characteristics of the Azure Kinect sensors. For the extrinsic calibration, we estimate the 6-DoF relative poses between cameras by performing stereo calibration with OpenCV [5] on pair-wise cameras. The calibration target is an asymmetric ChArUco target [2] to ensure unique keypoint labeling. Our dataset consists of undistorted camera RGBD images, where lens distortions are corrected. We also provide the corresponding optimal intrinsic parameters that best fit this undistorted geometry projection.

To quantitatively evaluate our calibration accuracy, we compute the reprojection error by projecting all the 3D calibration target points (board pattern corners) into the 2D image plane using our estimated camera parameters [32]. We then calculate the L2 norm between the reprojected points and the ground truth image observations. Throughout our five different scene setups, we achieve an average of 0.2 reprojection error (sub-pixel accuracy) for camera intrinsics parameters, and an average of 1.3 reprojection error (around one pixel offset) for extrinsics calibration.

After the camera calibration, we use Apriltags [37] to align the camera coordinate system with the photogrammetry background capture. This enables us to accurately overlay our multi-view camera capture with a high-resolution textured mesh model.

Synchronization. To ensure temporal alignment across multiple views, we synchronize the four Azure Kinect cameras using a daisy-chain configuration via 3.5 mm sync cables, following Microsoft's recommended protocol [3]. In this setup, one device acts as the master (or primary sensor), generating a synchronization signal that is relayed to the remaining cameras (subordinate sensors) through chained sync ports.

When using multiple Azure Kinect devices with overlapping fields of view, depth interference may occur due to simultaneous IR laser pulses. To avoid this, we follow Microsoft's protocol [3] and offset each subordinate device's depth capture start time by at least 160 µs relative to the others.

3.2 Geometry Processing

After acquisition, raw RGB and depth data are saved as lossless `.png` image files. Depth images are stored as 16-bit single-channel PNGs (in millimeters) to preserve precision and range. Using this RGB-D data, we generate multiple forms of 3D geometry representations. All geometry processing is performed using Open3D v0.19.0 [45] on an RTX 4090 GPU.

Point Cloud. We convert each depth map to a 3D point cloud by projecting depth pixels into 3D space using our intrinsic parameters of the corresponding RGB camera. These per-view (colored) point clouds are then transformed into a common world coordinate system using our extrinsic parameters obtained during calibration.

Textured Mesh. We generate textured meshes through a two-stage process: (1) 3D surface reconstruction via volumetric fusion, and (2) occlusion-aware multi-view texture mapping.

Reconstruction using TSDF. We reconstruct a mesh surface from multi-view depth images using Truncated Signed Distance Function (TSDF) fusion, implemented with Open3D's `VoxelBlockGrid`. TSDF is well-suited for RGB-D data, as it reduces sensor noise through spatial averaging, produces smooth and mesh surfaces, and handles overlapping views robustly.

We set the voxel size to 5 mm and exclude depth values beyond 10 m from integration to focus reconstruction within the effective sensing range. The truncation distance is set to 4 cm (eight times the voxel size), limiting TSDF updates to within ±4 cm of observed surfaces and reducing noise from distant or irrelevant regions. Our toolkit allows users to adjust these parameters to generate custom mesh reconstructions.

Occlusion-aware Texture Mapping. In multi-view RGB-D setups, naïve texture projection often causes ghosting, misalignment, or occlusion artifacts due to inconsistent or partially visible surfaces. To mitigate these issues, we employ a view-aware UV mapping approach that assigns each mesh triangle to the most geometrically consistent camera view based on per-vertex depth agreement.

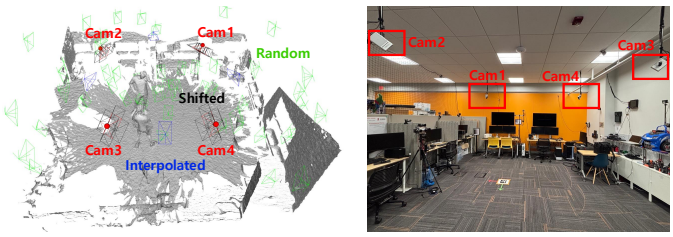


Fig. 2: Camera pose visualization in the capture room. Original (cam1–4), Interpolated, Shifted, Random viewpoint.

Each triangle is projected into all camera views, and visibility is estimated using a z-error metric—the absolute difference between the projected vertex depth and the corresponding value in the depth map. The triangle is then assigned to the view with the lowest aggregated z-error, and UV coordinates are computed with respect to that camera's RGB image. These UVs are normalized over a stitched texture atlas containing all RGB views. The final UVs are baked into the mesh representation, enabling texture sampling from reliable viewpoints and reducing common multi-view artifacts.

Gaussian Splat. We adopt the official implementation of 3D Gaussian Splatting [21], but instead of initializing with a sparse point cloud generated via Structure-from-Motion (SfM) over RGB images (as done in the original pipeline), we use high-fidelity point clouds directly obtained from depth sensors. This avoids the sparsity and misalignment commonly found in SfM-based reconstructions.

In detail, we use 4-band spherical harmonics (SH) and follow the original training schedule (30K iterations), where the Gaussian densification process begins at 0.5K iterations and terminates at 15K. With accurate multi-view RGB-D alignment and known camera poses, we supervise the splatting renderer by minimizing the difference between rendered and ground-truth RGB images.

The comparison of the different geometries is provided in Tab. 5, with further analysis discussed in § 4.2.

3.3 Camera Pose Variants

Unlike conventional 2D video, the perceptual quality of 3D content heavily depends on the viewer's 6-DoF perspective. Evaluating solely from the original, on-axis camera poses can introduce bias and overlook view-dependent artifacts such as occlusions, shading inconsistencies, or geometric distortions. To enable robust evaluation of volumetric data, particularly for tasks such as compression benchmarking, quality-aware metrics, and view synthesis, we provide multiple types of virtual camera poses per scene. An overview of the camera pose types is visualized in Fig. 2. Specifically, for each frame we generate:

- **Original views (4):** Calibrated camera extrinsics directly from the capture setup.
- **Shifted views (16):** Each original view is shifted in four directions (i.e., up, down, left, right) by 10 cm in the local camera frame.
- **Interpolated views (4):** Novel views are created by interpolating between consecutive original views. For each pair, we apply Spherical Linear Interpolation (Slerp) to the camera rotation and linear interpolation to the translation vector, generating midpoints in pose space.
- **Random views (50):** Cameras are sampled uniformly inside the scene's axis-aligned bounding box, each looking toward the scene center.

In total, each scene is rendered from **74 diverse viewpoints**. These views are included in our dataset to support view-aware evaluation pipelines and are used in our proposed quality-aware metric (§ 4.1). We also provide a script to generate additional views.

4 SceneHub4D Description & Analysis

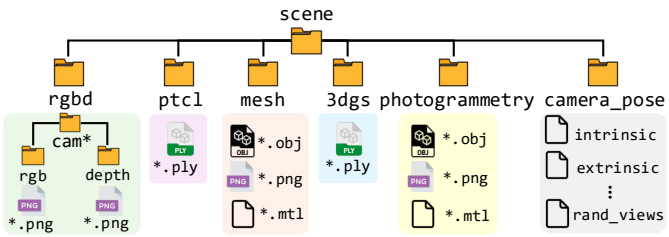


Fig. 3: Illustration of the dataset structure.

Dataset structure. Fig. 3 illustrates the structure of our dataset. For each scene, we provide multiple 3D representations with corresponding camera pose annotations. RGB and depth images are stored under `rgbd/cam*` directories, where each frame begins from index `1.png`. Due to the large data size, we provide only the first frame for geometry-based formats (e.g., point clouds, meshes, and Gaussian splats) as representative examples. The rest of the frames can be generated using the provided software tools. Camera poses are stored under the `camera_pose/` folder and include intrinsic and extrinsic parameters for each camera. All poses are defined in a world-to-camera coordinate convention and follow the Open3D coordinate system (right-handed with +Z forward, +Y up).

The dataset is hosted on the Open Storage Network (OSN) [31], an NSF-funded distributed data sharing and transfer service based on S3 storage. Since OSN is connected to Internet2, institutions on Internet2 can take advantage of high-speed data transfers for efficient access.

Dataset access, storage, and practical usage. Tab. 2 summarizes the storage footprint of the dataset. The full dataset, including all RGB-D frames across all scenes, photogrammetry assets, camera poses, and representative geometry files, occupies approximately 1.03 TB. To support rapid experimentation and evaluation, we additionally provide a lightweight subset that includes 100 RGB-D frames per scene together with camera poses and photogrammetry, requiring only 31.4 GB. This subset is designed for tasks such as compression benchmarking, reconstruction quality evaluation, and rapid experimentation, without requiring users to download the entire dataset.

Tab. 3 reports measured download performance from the Open Storage Network (OSN) for a representative subset (arena_scene1_100, ∼2 GB). At a local site connected via Internet2 (CMU, USA), the subset can be downloaded in approximately 40 seconds, while a geographically distant site (Inha University, South Korea) requires roughly 7 minutes at the measured average throughput.

Software suite. We provide a software tool implemented in Python and C++. It includes utilities to convert RGB-D streams into various 3D representations, instructions for generating Gaussian splats, and evaluation scripts for computing geometric and rendering quality metrics. In addition, we provide a lightweight Unity-based player and a web-based 3D viewer. The software and dataset are publicly available at `https://scenehub4d.github.io/`.

4.1 Metrics

To describe our dataset, we present a set of quantitative metrics to characterize the spatial, temporal, and geometric complexity of 3D dynamic scenes, accounting for both perceptual quality and structural detail.

Geometry Complexity Score (GCS). To quantify the geometric complexity, we define the *Geometry Complexity Score (GCS)* as:

$$\text{GCS} = \underbrace{\left(\frac{N_{\text{tri}}}{A_{\text{total}}}\right)}_{\text{Triangle density}} \times \underbrace{\left(1+\lambda_1 \cdot \frac{\sigma_{\text{area}}}{\mu_{\text{area}}} + \lambda_2 \cdot \frac{\sigma_\theta}{\mu_\theta}\right)}_{\text{Irregularity term}} \times \underbrace{\log(1+A_{\text{total}})}_{\text{Global scale weight}} \quad (1)$$

where N_{tri} is the total number of triangles and A_{total} is the total surface area (cm^2) of the mesh. The first term, $\frac{N_{\text{tri}}}{A_{\text{total}}}$, captures the *local triangle density*, measuring how densely the mesh is tessellated per unit area ($1/cm^2$). The second term reflects the *structural irregularity*, incorporating the coefficient of variation of triangle areas ($\frac{\sigma_{\text{area}}}{\mu_{\text{area}}}$), and the coefficient of variation of dihedral angles ($\frac{\sigma_\theta}{\mu_\theta}$), which accounts for variations in shape and curvature, respectively. This ensures that overly flat or uniformly tessellated geometry is not overestimated in complexity. The final term, $\log(1+A_{\text{total}})$, introduces a *scale-aware weighting* by modestly amplifying the score for larger surfaces. Without this, two meshes with identical density and local variation (e.g., a small object and a large room) could receive similar scores despite vastly different spatial scopes. The logarithmic form prevents disproportionate scaling for extremely large meshes. We set both λ_1 and λ_2 to 0.5 by default. In § 4.2, we further analyze the behavior of this metric under geometry simplification, illustrating how it reflects structural reduction and supports quality–complexity trade-off analysis.

Component	Data Size
RGB-D data (all frames, all scenes)	956.9 GB
RGB-D data (100 frames per scene)	28.6 GB
Photogrammetry	2.7 GB
Camera poses	73.7 KB
Geometry (first frames only)	40.9 GB
Total (full dataset)	**1.03 TB**
Total (lightweight subset)†	**31.4 GB**

Table 2: Storage requirements of the SceneHub4D dataset. †The lightweight subset includes RGB-D data with 100 frames per scene, along with camera poses and photogrammetry, and is intended for rapid testing and evaluation.

Measurement Site	Avg. Download Speed (Mbps)
CMU (USA)	410.5 ± 8.6
Inha Univ. (South Korea)	37.2 ± 4.1

Table 3: Download performance from OSN storage for a representative scene subset (arena_scene1_100). It consists of 100 RGB-D frames (2 GB). At the measured speeds, downloading the subset takes approximately 40 seconds from a local site and 7 minutes from a geographically distant site.

Multi-view SSIM. To quantify view-dependent perceptual quality, we compute the average SSIM across sets of camera poses of different types: original (`orig`), shifted (`shft`), interpolated (`intp`), and random (`rndm`) views, as elaborated in § 3.3. Each camera pose $\mathbf{P}_i$ is represented as a 3×4 extrinsic matrix composed of a 3×3 rotation matrix $\mathbf{R}_i$ and a 3D translation vector $\mathbf{t}_i$, i.e., $\mathbf{P}_i = [\mathbf{R}_i \mid \mathbf{t}_i]$. This matrix transforms world coordinates into the camera coordinate frame and defines the viewpoint from which the scene is rendered.

Let $\mathscr{P}_v$ denote the set of poses of type v, and $N_v = |\mathscr{P}_v|$. For each pose type $v \in \{\texttt{orig}, \texttt{shft}, \texttt{intp}, \texttt{rndm}\}$, we define the multi-view SSIM as:

$$\text{SSIM}^{(v)}_{\text{multi}} = \frac{1}{N_v} \sum_{\mathbf{P}_i \in \mathscr{P}_v} \text{SSIM}(R(\mathbf{P}_i), G(\mathbf{P}_i)), \quad (2)$$

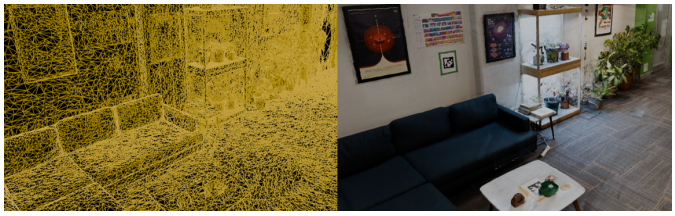
where $R(\mathbf{P}_i)$ and $G(\mathbf{P}_i)$ denote the rendered and ground-truth images at pose $\mathbf{P}_i$, respectively.

Quality-aware GCS. While the raw geometry reconstructed from capture data often contains a very large number of triangles and therefore exhibits a high Geometry Complexity Score (GCS), such complexity may be unnecessary depending on the application's tolerance for perceptual quality loss. For example, if surface details are visually preserved through textures, minor geometric simplifications may go unnoticed, as shown in Fig. 4. To reflect this, in addition, we define a quality-aware GCS that considers the trade-off between geometric complexity and perceptual fidelity.

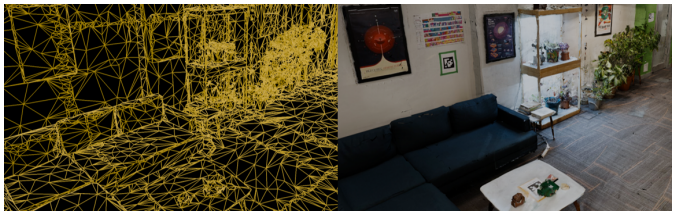
Specifically, we apply mesh simplification based on the Quadric Error Metric [10] at various decimation ratios (50%, 60%, ..., 95%,

Scene / Coverage	Activity	ID	Actors	Frames	$\frac{N_{\text{tri}}}{A_{\text{total}}}$ ($\frac{1}{\text{cm}^2}$)	GCS	$\text{GCS}_{\text{SSIM}\geq 0.98}$	Depth SI	V-TI (mm)
Lab area 6.61 m×8.52 m×2.96 m	Conversation, vacuuming, walking	1	2	2549	11.75	105.6	21.7	2566	6.18
	Baseball batting practice	2	4	2569	11.71	104.7	21.1	2547	6.73
	Group hangout	3	4	2695	11.77	105.86	32	2564	7.16
	Yoga, dancing	4	1	2412	11.66	104.6	21.16	2656	5.43
	Moving chair, navigation	5	1	1753	11.72	105.2	21.31	2630	6.19
	VR gameplay using Quest	6	1	1713	11.76	106.1	21.42	2643	5.53
	Static background	0	–	8	11.69	105.6	21.34	2560	3.96
Couch 9.99 m×8.35 m×3.54 m	Reading and talking	1	2	5253	11.73	101.93	20.43	1767	7.14
	Informal hangout	2	3	5266	11.75	101.94	21.7	1762	7.33
	Fixing a toy	3	3	5249	11.78	101.49	20.39	1718	6.89
	Static background	0	–	8	11.64	100.83	20.25	1657	3.36
Kitchen 5.43 m×5.24 m×3.3 m	Talking, searching for items	1	1	5344	12.21	105.38	30.83	1627	6.47
	Making coffee	2	2	5327	12.19	104.86	32	1631	6.74
	Group talk	3	3	4994	12.28	104.72	30.98	1728	6.07
	Static background	0	–	8	12.23	105.78	30.84	1680	3.25
Hallway/Whiteboard 11.92 m×6.96 m×4.44 m	Writing on board	1	2	4914	11.84	107.61	32	3043	6.75
	Presenting with gestures	2	3	3900	11.81	106.54	21.69	3037	8.24
	Presenting (alternate angle)	3	2	3000	11.79	106.55	32	3045	7.36
	Static background	0	–	8	11.84	108.48	21.7	3105	4.25
Factory 12.62 m×8.77 m×4.33 m	Conversation, walking	1	2	2735	12.14	108.2	32.06	2748	7.00
	Searching for items	2	2	2758	12.26	108.23	31.9	2912	7.86
	Static background	0	–	8	12.16	108.18	31.95	3113	5.0

Table 4: Dataset description with scene size, actor count, frame count, triangle density, *GCS*, $\text{GCS}_{\text{SSIM}\geq 0.98}$, Depth SI, and V-TI.



(a) Original scene (GCS = 100.19 / Multi-view SSIM = 1)



(b) Decimated scene (GCS = 11.1 / Multi-view SSIM = 0.98)

Fig. 4: Example comparison of original and decimated scenes. Decimation significantly reduces geometric complexity (GCS) with only marginal loss in perceptual quality (e.g., multi-view SSIM). Wireframe renderings (left image) highlight geometric structure, while textured surface renderings (right image) illustrate visual appearance.

99%), and for each simplified mesh, we compute its multi-view SSIM as defined in Equation 2. Among these candidates, we select the mesh with the lowest geometric complexity that satisfies a target perceptual quality threshold, defined as SSIM $\geq \tau$. We use $\tau = 0.98$ by default, since SSIM values above 0.98 are generally considered perceptually indistinguishable from the reference. We refer to the resulting score as the *SSIM-aware GCS*, defined in display form as:

$$\text{GCS}_{\text{SSIM}\geq\tau} \tag{3}$$

This metric captures the effective geometric complexity required to preserve perceptual quality at a given level, and provides a more application-aware interpretation of geometric detail. While we use multi-view SSIM in our evaluation, other perceptual metrics (e.g., PSNR, LPIPS [43]) or geometric metrics (e.g., Chamfer distance, Hausdorff distance) can be substituted depending on the application case.

Spatial and (Volumetric) Temporal Information. We adopt the spatial and temporal information (SI and TI) metrics as defined in the ITU-T Recommendation P.910 [17], which are widely used in video quality assessment. For each scene, we compute SI and TI on both RGB and depth image sequences across all camera views, and report the average values, respectively. Spatial Information (SI) measures the amount of spatial detail in a frame, computed as the maximum standard deviation of the Sobel-filtered image (which detects intensity edges) across time. Temporal Information (TI) captures frame-to-frame variation and is computed as the standard deviation of pixel-wise temporal differences between consecutive frames. The final TI is defined as the maximum of these standard deviations across the sequence.

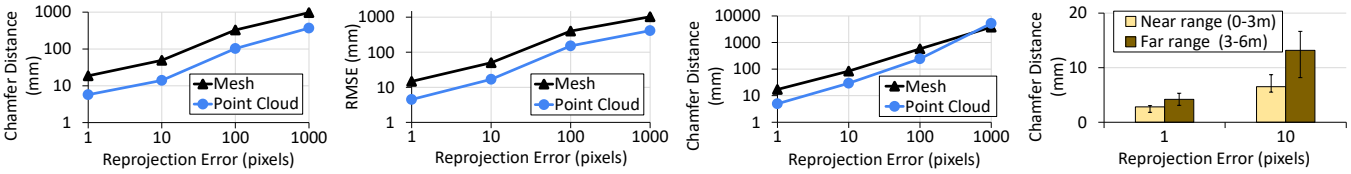
Traditional TI metrics do not transfer well to depth maps. Noise, occlusion, and inconsistent spatial scale often hinder meaningful motion analysis. Therefore, we reinterpret temporal information in a volumetric context by analyzing changes across successive 3D point clouds. Specifically, given a sequence of point clouds $\{P^{(t)} = \{\mathbf{x}_i^{(t)}\}\}_{t=1}^{T}$, we compute frame-wise correspondences $\mathscr{C}^{(t)}$ using Iterative Closest Point (ICP). Then, the **Volumetric Temporal Information (V-TI)** is defined as the maximum standard deviation of 3D displacements over the sequence.

$$\text{V-TI} = \max_t \sigma\left(\left\|\mathbf{x}_i^{(t)} - \mathbf{x}_i^{(t+1)}\right\|_2\right), \quad \forall(\mathbf{x}_i^{(t)}, \mathbf{x}_i^{(t+1)}) \in \mathscr{C}^{(t)} \tag{4}$$

where $\sigma(\cdot)$ denotes the standard deviation across all matched points in each frame pair. This metric captures geometric deformation over time.

Geometry Fidelity Metrics. In addition to perceptual quality, we assess geometry fidelity using two standard measures in our compression benchmark (§ 5): Chamfer Distance (CD) and Root Mean Squared Error (RMSE). Given two point sets $P = \{p_i\}_{i=1}^{N}$ and $Q = \{q_j\}_{j=1}^{M}$, the symmetric Chamfer Distance is defined as:

$$\text{CD}(P,Q) = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|_2^2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|q-p\|_2^2. \tag{5}$$



(a) Chamfer distance vs. reprojection error on camera extrinsics. **(b)** RMSE vs. reprojection error on camera extrinsics. **(c)** Chamfer distance vs. reprojection error on camera intrinsics. **(d)** Geometry error across depth ranges under reprojection error on camera extrinsics.

Fig. 5: Effect of calibration errors on geometry reconstruction accuracy, measured by Chamfer distance and RMSE.

This metric captures the average bidirectional deviation between two shapes. RMSE is computed as the root mean squared Euclidean error between corresponding 3D points extracted from mesh vertices after alignment.

4.2 Analysis

Impact of calibration errors on geometry reconstruction. To better understand how calibration errors propagate through our reconstruction pipeline, we analyze the sensitivity of different geometry representations under controlled reprojection perturbations on both camera extrinsics and intrinsics (Fig. 5). Since obtaining error-free ground-truth geometry for room-scale scenes is challenging in practice, we treat reconstructions generated using our measured calibration parameters as a reference and examine how geometry quality degrades as calibration parameters are progressively perturbed. Specifically, we inject synthetic reprojection errors ranging from 1 to 1000 pixels into the calibrated camera parameters and measure the resulting deviation from the reference geometry using Chamfer distance and RMSE.

For **extrinsic** perturbations, our analysis suggests that the calibration accuracy observed in our setup (approximately 1.3-pixel reprojection error) leads to only single-digit millimeter geometry deviation for point cloud (< 5 mm). As reprojection error increases beyond this measured regime, both Chamfer distance and RMSE grow monotonically (Fig. 5(a–b)), indicating that pose misalignments increasingly accumulate during multi-view fusion. Across the full perturbation range, mesh-based reconstructions are more sensitive than point clouds, as surface fusion amplifies inter-view misalignment through TSDF integration.

For **intrinsic** perturbations, we observe a similar pattern: the calibration accuracy observed in our calibration (sub-pixel reprojection error) is associated with stable geometry across representations, while larger intrinsic errors lead to measurable distortions (Fig. 5(c)).

To further characterize how such errors manifest across scene depth, Fig. 5(d) breaks down Chamfer distance into near (0–3 m) and far (3–6 m) ranges under extrinsic perturbations. The far range incurs higher error than the near range at the same reprojection offset, indicating that calibration errors disproportionately affect distant surfaces from the camera.

Overall, Fig. 5 shows that calibration errors propagate through the pipeline in a bounded and interpretable manner. Based on this perturbation analysis, our measured calibration accuracy (sub-pixel intrinsics and approximately 1-pixel extrinsic reprojection error) appears sufficient to maintain stable geometry quality for both point clouds and meshes, and is compatible with reliable initialization of derived representations such as Gaussian splats. Practically, moderate calibration inaccuracies tend to impact (i) fused mesh reconstructions more than per-frame point clouds and (ii) far-range geometry more than near-field surfaces.

Quantitative characterization of the dataset using the proposed metrics. Tab. 4 summarizes key characteristics of our dataset using the proposed metrics (§ 4.1). Each scene spans a volumetric coverage ranging from approximately $94 - 480$ m^3, capturing a diverse set of environments, and contains on average over 2,500 frames per activity sequence. In addition to reporting raw GCS, we include a perceptual variant, $\text{GCS}_{\text{SSIM}\geq 0.98}$, to reflect the amount of geometry that can be simplified without compromising visual quality, as measured by SSIM. For example, the "Lab area" scenes exhibit a GCS of around 105.6

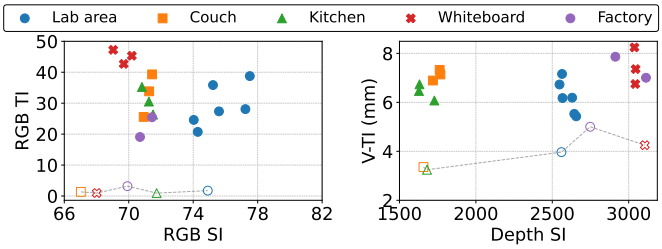


Fig. 6: SI & TI distribution of our dataset. Hollow markers and dotted lines denote static background sequences.

but can be reduced to a perceptual GCS of 21.7–32.0 while preserving SSIM ≥ 0.98. This illustrates that despite high initial complexity, substantial simplification is achievable with minimal perceptual loss.

We also evaluate spatial and temporal dynamics using Depth SI and Volumetric Temporal Information (V-TI). Interestingly, even in static background scenes, V-TI values remain non-zero (e.g., 3.25–5.0 mm), primarily due to depth sensor noise and sporadic miscaptures (e.g., floating holes) across frames. However, the distinction between dynamic and static scenes is evident—for instance, dynamic scenes such as "Presenting with gestures" exhibit V-TI up to 8.24 mm, representing over $2\times$ the movement magnitude compared to static backgrounds.

Fig. 6 visualizes the distribution of Spatial and Temporal Information across scenes. The RGB SI–TI plot (left) reveals that scenes cluster naturally based on motion and scene content. Hollow markers and dotted lines denote static background sequences, which cluster distinctly apart from dynamic scenes. Similarly, in the Depth SI vs. V-TI plot (right), we observe a clear separation: factory and whiteboard scenes show high Depth SI and V-TI, indicating richer motion and geometric variation, whereas static backgrounds occupy the lower end of the spectrum. These trends validate that our metrics are effective in characterizing both spatial richness and dynamic behavior in 4D scenes, and further demonstrate the diversity of real-world captures with varying geometric and motion characteristics.

Comparison with different datasets. As shown in Fig. 7, our dataset captures (even without the photogrammetry background) significantly larger volumes, covering full background environments beyond the primary subjects. While Panoptic [20] includes background scenes, it relies on older Kinect v2 sensors and a single fixed dome setup, limiting both resolution and spatial extent. PCVD [25] and CWIPC [33] use modern Azure Kinect sensors to produce denser point clouds, but are restricted to small indoor volumes centered on human activity. Our dataset combines the strengths of both: high-resolution depth sensing and large-scale, full-scene coverage. Our scenes exhibit the highest point density, as we cover expansive backgrounds using modern RGB-D sensors.

Among the compared datasets, only PCVD provides 3D meshes, allowing a mesh-level comparison shown in Tab. 6. Unsurprisingly, our dataset contains significantly more triangles and surface area due to broader scene coverage. The dihedral angle distributions also differ, reflecting the structural complexity of the scenes. In terms of geometry complexity, our GCS is over $2.5\times$ higher than PCVD.

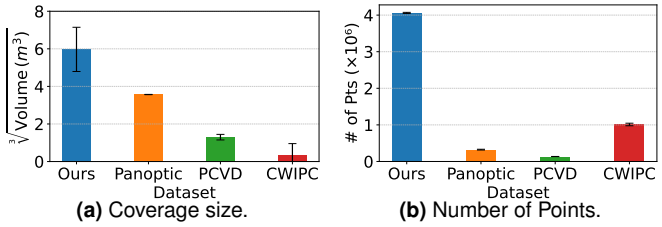
Comparison of different geometry representations. Over the years, the computer vision and graphics community has developed diverse formats for representing 3D scenes, ranging from raw per-view RGB-D

Representation	Components	Data Type	Shape	Size (MB)	Total Size
RGB-D ($N_{\text{cam}} = 4, H = 1080, W = 1920$)	Depth	uint16	$(H, W, 1, N_{\text{cam}})$	16.6 (40%)	**41.48 MB**
	RGB	uint8	$(H, W, 3, N_{\text{cam}})$	24.88 (60%)	
Point Cloud (P = 4.2 M)	Points (x,y,z)	float32	$(P, 3)$	50.44 (80%)	**63.05 MB**
	Colors (r,g,b)	uint8	$(P, 3)$	12.61 (20%)	
Textured Mesh (T = 1.3 M, V = 3.91 M)	Vertices	float32	$(V, 3)$	46.96 (28.3%)	**165.82 MB**
	Triangles	int32	$(T, 3)$	15.67 (9.5%)	
	UV Coordinate	float32	$(V, 2)$	31.35 (18.9%)	
	Normals	float32	$(V, 3)$	46.96 (28.3%)	
	Texture	uint8	$(H, W, 3, N_{\text{cam}})$	24.88 (15%)	
Gaussian Splat (G = 4.4 M)	Positions	float32	$(G, 3)$	52.87 (5.2%)	**1.005 GB**
	Scales	float32	$(G, 3)$	17.62 (1.8%)	
	Rotations	float32	$(G, 4)$	70.49 (7%)	
	Opacities	float32	$(G, 1)$	17.62 (1.8%)	
	SH coefficients (degree 0)	float32	$(G, 3, 1)$	52.87 (5.3%)	
	SH coefficients (degree 1–3)	float32	$(G, 3, 15)$	793.02 (78.9%)	

Table 5: Raw data size comparison across explicit 3D representations for one temporal frame t (Geometry, Color component). N_{cam}, P, V, T, and G denote the number of camera views, points, vertices, triangles, and gaussians, respectively.

Dataset	Triangle Count (M)	Surface Area (m^2)	Dihedral Angle (°)	Coverage ($\sqrt[3]{m^3}$)	GCS
SceneHub4D	1.5 (±0.6)	45.5 (±14.3)	76.3 (±3.8)	6 (±1.2)	104.94 (±2.1)
PCVD [25]	0.17 (±0.02)	2.58 (±0.4)	91.12 (±0.6)	1.3 (±0.1)	41.84 (±1.1)

Table 6: Comparison of mesh statistics with different dataset: SceneHub4D exhibits significantly larger triangle count and surface area due to broader scene coverage, resulting in a GCS over 2.5× higher than PCVD.



(a) Coverage size. **(b)** Number of Points.

Fig. 7: Dataset comparison in point cloud.

images to point clouds, triangle meshes, and most recently, Gaussian Splat representations. Our dataset enables direct comparison across these multiple 3D representations under identical scenes. As summarized in Table 5, we analyze the raw data footprint of each format to clarify differences in structure and aid representation choices for immersive applications.

In detail, the raw RGB-D images (N_{cam}=4 views, 1920×1080) contain about 1.02 M valid depth pixels per frame (roughly 50% coverage), which correspond to 4.2 M points after projection and fusion. This results in a point cloud requiring 63.05 MB. From this, a textured mesh is reconstructed using TSDF fusion, producing 3.91 M vertices and 1.3 M triangles, and expanding the size to 165.82 MB due to the addition of normals and texture coordinates.

The Gaussian Splat representation, initialized from the same point cloud, results in a larger storage footprint of approximately 1.0 GB due to gaussian densification during training (P=4.2 M → G=4.4 M) and high-dimensional attributes (e.g., 793.02 MB for Spherical Harmonics (SH) coefficients).

Our dataset and analysis help convey an intuitive sense of how different 3D representations compare in practice. For instance, representing the same scene with 3D Gaussian Splatting requires approximately 6–16× more storage than traditional formats such as meshes and point clouds, due to densification and high-dimensional attributes. In addition to providing all representations, we offer conversion tools and precise pose metadata, enabling reproducible, structured analysis across formats. We hope this helps researchers and developers better understand trade-offs in representation choices when building 3D-enabled systems.

4.3 Geometry Complexity Score Deep Dive

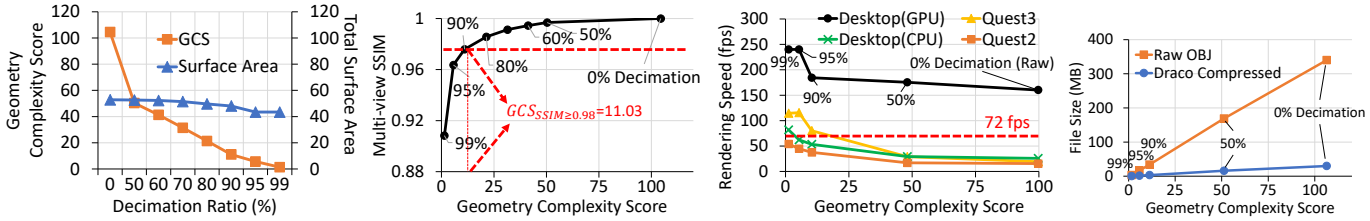
An effective complexity metric should serve as a practical proxy for capturing the richness of 3D geometry in a way that correlates with perceptual quality, rendering cost, and storage footprint. In this section, we investigate whether the proposed Geometry Complexity Score (GCS) satisfies this role through a series of targeted experiments. Specifically, we ask:

- **Q1:** How does GCS behave under progressive geometry simplification? Does it effectively capture reductions in complexity while preserving overall 3D structure?
- **Q2:** How does GCS correlate with perceptual quality?
- **Q3:** How does rendering performance vary with GCS, particularly on compute-constrained VR devices?
- **Q4:** How does GCS correlate with actual storage size?

GCS behavior and implication (A1). To validate the effectiveness of the proposed GCS, we analyze how it behaves under geometry simplification. Specifically, we perform progressive mesh simplification based on the Quadric Error Metric [10] at various triangle decimation ratios from 0% (original mesh) to 99% (heavily simplified geometry), and observe how GCS responds.[1]

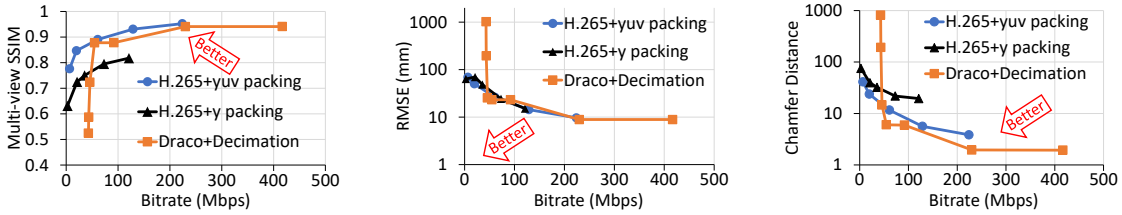
As shown in Fig. 8a, GCS decreases with increasing decimation. This trend is intuitive: as the geometry becomes simpler, the number of polygons reduces significantly while the overall surface area remains relatively stable. This implies that the high-level structure and topology of the 3D scene are preserved, even as local geometric details are reduced. The total surface area curve confirms this stability. GCS effectively captures this behavior by reflecting both the reduction in triangle density and the preservation of global structure.

[1] Here, we use decimation and simplification interchangeably.



(a) GCS vs. geometry simplification. **(b)** GCS vs. perceptual quality. **(c)** GCS vs. rendering speed (fps). **(d)** GCS vs. actual file size.

Fig. 8: GCS Deep Dive: Analyzing its relationship with geometry simplification, perceptual quality, rendering speed, and data size.



(a) Multi-view SSIM vs. bitrate. **(b)** RMSE vs. bitrate. **(c)** Chamfer distance vs. bitrate.

Fig. 9: Volumetric video compression benchmark results using SceneHub4D dataset and metrics.

GCS vs. Perceptual Quality (A2). Fig. 8b shows how perceptual quality, measured via our proposed multi-view SSIM (§ 4.1) using all the camera poses (§ 3.3), evolves with respect to GCS. As the geometry is progressively simplified (i.e., GCS decreases), the multi-view SSIM remains largely stable, staying above 0.95 even under high decimation ratios ($> 80\%$, i.e., maintaining less than 20% of the geometry). This sublinear degradation shows that substantial reductions in geometric complexity can be achieved without noticeable loss in visual quality. This relationship aligns with human visual perception, which is typically more sensitive to texture and lighting changes than to geometric detail reduction, as shown in Fig. 4.

Motivated by this observation, we introduced a perceptual quality threshold (e.g., SSIM ≥ 0.98), and we defined a quality-aware complexity score, $\text{GCS}_{\text{SSIM}\geq\tau}$. This is the minimal GCS that satisfies the given quality constraint. In our example, $\text{GCS}_{\text{SSIM}\geq 0.98} = 11.03$, as indicated by the red dashed lines and annotation in the figure.

GCS vs. Rendering Speed (A3). To evaluate how geometric complexity, as quantified by GCS, impacts rendering performance, we measured the frame rates of our captured scenes on devices with varying compute capabilities. Specifically, we tested on two standalone VR headsets, Meta Quest 2 and Meta Quest 3 as well as a desktop machine equipped with an Intel i9-13900K CPU and NVIDIA RTX 4090 GPU. Each VR headset was configured to target a maximum display refresh rate of 120 Hz.

As shown in Fig. 8c, rendering speed decreases as GCS increases, confirming the expected trend: more complex geometry leads to higher shading and rasterization load. This relationship is observed across all devices. Furthermore, for a given GCS, devices with stronger compute capabilities achieve higher frame rates; Quest 3 and the desktop GPU maintain significantly better performance than Quest 2 or a CPU-only renderer using OpenGL.

In addition to confirming the relationship between GCS and rendering speed, these findings provide concrete guidelines for how our dataset can be rendered on different VR platforms under real-time constraints. When considering real-time rendering thresholds, such as the 72 FPS baseline (indicated by the red line) commonly used in VR applications, we find that both the desktop GPU and Quest 3 can maintain real-time performance up to approximately 10% of the original mesh (i.e., 90% decimation) for our captured scenes. As shown in Fig. 8b, this level of simplification still preserves high perceptual quality (SSIM ≥ 0.98), suggesting that local rendering at reduced complexity is viable on these devices.

In contrast, for less capable devices such as desktop CPU-only setups or Quest 2, real-time rendering is not achievable even with significant simplification. In these cases, remote rendering on a powerful GPU, followed by image-based streaming to the client, should be preferable to maintain responsiveness and visual quality.

GCS vs. File Size (A4). Finally, we investigate how GCS relates to storage size. As seen in Fig. 8d, file size scales approximately linearly with GCS. In particular, when comparing raw OBJ files to Google Draco [12] (lossless) compressed files, we observe an average reduction of about $10.25\times$ in file size but the overall trend remains consistent across both formats. The highest compression ratio ($\sim 11\times$) occurs at the largest complexity (GCS $= 106$), while the ratio gradually decreases to around $9\times$ at the lowest complexity level (GCS $= 1.5$).

The result shows that GCS correlates linearly with storage size, indicating that it effectively captures the actual complexity of the geometry in both raw and compressed representations.

Across all four dimensions (i.e., structure, quality, rendering speed, and storage), GCS exhibits consistent and interpretable behavior. This highlights the role of GCS as a practical abstraction for geometry scene complexity.

5 SceneHub4D Use Cases

One important application of SceneHub4D is research on live volumetric video streaming and compression. Datasets for this purpose must provide realistic content (sensor noise, room-scale, long-duration captures), be directly usable for replay and integration into streaming pipelines, and enable reproducibility without complex custom setups. Due to the lack of publicly accessible, full-scene volumetric video datasets, prior works on volumetric streaming have often relied on closed custom datasets with their own camera setups [13, 14, 19, 23], lower-resolution Panoptic captures [24, 42], or composited full-scene content constructed from separate foreground and background assets [35, 36].

SceneHub4D allows researchers to reproduce live-like sequences directly from raw captures, eliminating the burden of complex calibration and synchronization. As a publicly available dataset captured with modern high-resolution sensors, it lowers the barrier to entry for research and ensures broad reproducibility. Moreover, it provides diverse, long video sequences with full-scene backgrounds, unlike prior datasets that focused mainly on foreground actors [25, 33]. These realistic backgrounds make SceneHub4D especially valuable for evaluating temporal compression, since inter-frame redundancy is most evident in static or slowly changing regions. In fact, several research groups are already employing SceneHub4D dataset in their compression studies, with early works currently under submission.

In the following, we present a lightweight benchmarking study illustrating its use in live volumetric streaming and compression.

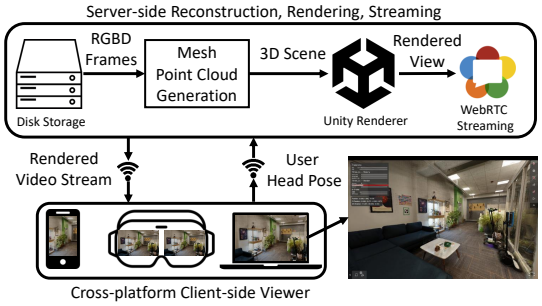


Fig. 10: SceneHub4D's player and streaming overview: Local replay and remote rendering pipeline for streaming volumetric content to VR clients.

Compression benchmark. We focus on live compression in real-time streaming scenarios and select representative pipelines that are both widely used and directly compatible with our RGB-D captures. [2] To demonstrate this, we conduct a benchmarking study across three representative geometry compression pipelines: (1) mesh decimation followed by Draco encoding [12, 19], (2) image-based depth compression using H.265 on 8-bit quantized depth maps with Y-only packing [13, 22, 23], and (3) an enhanced variant that applies full YUV packing, providing more bit space for depth representation. For depth compression, we evaluate H.265 with QP = {5, 15, 25, 35, 45}. For the Draco + decimation approach, we consider 95% and 99% decimation levels, combined with Draco QP = {0(lossless), 2, 4, 8, 14} with the highest compression level (cl=10), and derive the Pareto-optimal curve for each decimation level. In all methods, texture images are encoded using H.265 with YUV420p pixel format at QP = 30. We assess perceptual quality using multi-view SSIM and geometry fidelity using Chamfer Distance and RMSE across a wide range of bitrates.

As shown in Figure 9, Draco (a 3D codec) preserves geometry well at high bitrates, but struggles to maintain visual quality under aggressive compression. Because it compresses each frame independently and cannot exploit inter-frame redundancy, making it difficult to achieve efficient compression at low bitrates (e.g., $<$100 Mbps) without significant quality degradation. In contrast, H.265-based depth compression (a 2D video codec) achieves better perceptual quality at lower bitrates thanks to inter-frame prediction, but still suffers from degraded geometric accuracy (Fig. 9c) compared to Draco. This degradation is primarily caused by 8-bit quantization, which limits depth precision, particularly in background regions with large depth variation. Our dataset helps reveal such limitations more clearly, especially in scenes with complex and temporally coherent backgrounds.

Motivated by our observation, we applied a simple modification to the H.265-based compression pipeline: replacing Y-only packing with full YUV packing, thereby leveraging all three channels (24 bits) for depth encoding. We refer to this variant as H.265+YUV packing in the figure, which directly replaces the original Y-only packing. This change significantly improved perceptual quality (multi-view SSIM) across all bitrates without increasing geometric error. This demonstrates how our dataset enables not only direct comparisons across compression pipelines, but also the discovery and validation of new optimizations. For example, advanced mesh compression methods [7] can use this dataset to evaluate their performance.

Remote rendering and streaming. To support research in volumetric video streaming, we developed a reference playback system that replays captured sequences from our dataset in a "live-like" manner, simulating real-time capture and transmission. The pipeline is shown in Figure 10. Specifically, we implemented a Unity-based local player that renders the preprocessed frames as if they were being streamed live, enabling controlled replay and view navigation.

In addition, we developed a remote rendering platform that transmits camera poses and rendering requests to a server, which performs the rendering and streams the resulting rendered views as a video stream to client devices, including VR headsets, phones, and desktop computers. This setup allows researchers to evaluate bandwidth, latency, and quality trade-offs between local and cloud-based rendering under varying levels of geometry complexity (e.g., different GCS levels).

Our dataset thus supports both offline evaluation and real-time end-to-end testing for volumetric video delivery systems, bridging the gap between capture, rendering, and streaming in immersive applications [15, 27, 28, 41].

[2] While MPEG V-PCC and G-PCC are well-known geometry codecs, they are not directly applicable to RGB-D live capture and are computationally too heavy for real-time operation in live streaming scenarios.

6 Limitations & Future Directions

SceneHub4D focuses on room-scale indoor environments captured using a multi-view RGB-D setup, and thus has several inherent limitations. First, our captures do not directly generalize to outdoor scenes or extreme lighting conditions, as the depth sensors rely on active infrared projection and operate reliably only within a limited range (0.5–6 m) [30]. Very large spaces or very small confined environments may therefore exhibit reduced depth quality or incomplete coverage.

Second, our current setup uses four fixed cameras in a static studio configuration, which limits viewpoint coverage and can lead to occlusions and missing depth regions. To partially mitigate this, we incorporate a high-resolution photogrammetric background reconstructed from a large number of RGB images, which helps fill geometric gaps outside the effective depth sensing range. Nevertheless, raw RGB-D data may still contain sensor noise, depth holes, and inaccuracies, particularly near object boundaries and reflective surfaces.

These limitations suggest several directions for future extensions. Advances in learning-based depth completion [26, 34], estimation [38, 39] and multi-view stereo reconstruction [6, 18, 40] could be integrated to improve depth quality and recover missing geometry beyond the physical sensing limits. Increasing the number of cameras could further improve coverage.

Finally, the compression benchmarking presented in this paper is intended as a representative evaluation rather than an exhaustive comparison. Our analysis focuses on a small set of widely used compression pipelines that are practically deployable. A comprehensive comparison with emerging neural-based compression methods and live volumetric streaming approaches would require careful consideration of multiple factors, including evaluation metrics, encoding parameters, encoding and decoding efficiency, hardware requirements, and data format compatibility, which are beyond the scope of this paper. We leave such extensive comparisons for future work.

7 Conclusion

This paper presents SceneHub4D, a comprehensive dataset and evaluation framework for 6-DoF 4D VR scenes. Our dataset provides long-duration, full-scene RGB-D captures with high-quality photogrammetric backgrounds, multiple geometry representations, and diverse virtual camera poses for robust evaluation. Beyond static analysis, we proposed new metrics such as GCS and V-TI and validated their utility through rendering experiments on desktop GPUs and standalone VR headsets, highlighting device-specific trade-offs between complexity, quality, and frame rate. We further demonstrated how SceneHub4D can support practical applications, including compression benchmarking and live streaming, through a Unity-based player and remote rendering system. We envision SceneHub4D as a foundation for standardized evaluation in immersive media, bridging capture, representation, and delivery, and enabling future research in scalable volumetric video systems and VR applications.

Acknowledgments

We thank anonymous reviewers for providing helpful feedback and suggestions to improve our work. We also thank Ankush Jain and the WiseLab members at Carnegie Mellon University for their assistance with data collection. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. RS-2024-00406250), and by INHA UNIVERSITY Research Grant.

REFERENCES

[1] Azure Kinect DK. https://azure.microsoft.com/en-us/services/kinect-dk/, Mar. 2019. Online; accessed May 2025. 2

[2] Detection of ChArUco Boards. https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html, Mar. 2019. Online; accessed May 2025. 3

[3] Synchronize multiple Azure Kinect DK devices. https://learn.microsoft.com/en-us/previous-versions/azure/kinect-dk/multi-camera-sync, Feb. 2020. Online; accessed May 2025. 3

[4] Reality Capture. https://www.capturingreality.com/, May 2025. Online; accessed May 2025. 2

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 3

[6] Y. Cabon, L. Stoffl, L. Antsfeld, G. Csurka, B. Chidlovskii, J. Revaud, and V. Leroy. Must3r: Multi-view network for stereo 3d reconstruction. In *CVPR*, 2025. 9

[7] G. Chen, F. Hácha, L. Váša, and M. Dasari. Tvmc: Time-varying mesh compression using volume-tracked reference meshes. In *Proceedings of the 16th ACM Multimedia Systems Conference*, pp. 79–89, 2025. 9

[8] C. B. Duane. Close-range camera calibration. *Photogramm. Eng*, 37(8):855–866, 1971. 3

[9] FSVVD. Fsvvd project page. https://cuhksz-inml.github.io/full_scene_volumetric_video_dataset/, 2026. 2

[10] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 209–216, 1997. 4, 7

[11] G. Gautier, X. Zhou, T. Nguyen, J. Jansen, L. Fréneau, M. Viitanen, U. Phan, J. Käpylä, I. Viola, A. Mercat, P. Cesar, and J. Vanne. Uvg-cwi-dqpc: Dual-quality point cloud dataset for volumetric video applications. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, 7 pages, p. 13112–13118. Association for Computing Machinery, New York, NY, USA, 2025. doi: 10.1145/3746027.3758263 2

[12] Google. Draco 3d data compression library. https://google.github.io/draco/, 2024. 8, 9

[13] Y. Guan, X. Hou, N. Wu, B. Han, and T. Han. Metastream: Live volumetric content capture, creation, delivery, and rendering in real time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pp. 1–15, 2023. 8, 9

[14] B. Han, Y. Liu, and F. Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*, pp. 1–13, 2020. 8

[15] K. Hu, Y. Chen, K. Han, B. Li, H. Yang, Y. Jin, J. Liu, and F. Wang. Livevv: Human-centered live volumetric video streaming system. *IEEE Internet of Things Journal*, 12(11):17233–17243, 2025. doi: 10.1109/JIOT.2025.3535812 9

[16] K. Hu, Y. Jin, H. Yang, J. Liu, and F. Wang. Fsvvd: A dataset of full scene volumetric video. In *Proceedings of the 14th ACM Multimedia Systems Conference*, MMSys '23, 6 pages, p. 410–415. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3587819.3592551 2

[17] ITU-T Recommendation. Subjective video quality assessment methods for multimedia applications. Technical Report P.910, International Telecommunication Union, October 2023. Version 6.0. 5

[18] W. Jang, P. Weinzaepfel, V. Leroy, L. Agapito, and J. Revaud. Pow3r: Empowering unconstrained 3d reconstruction with camera and scene priors. In *CVPR*, 2025. 9

[19] T. Jin, M. Dasa, C. Smith, K. Apicharttrisorn, S. Seshan, and A. Rowe. Meshreduce: Scalable and bandwidth efficient 3d scene capture. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pp. 20–30. IEEE, 2024. 2, 8, 9

[20] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 6

[21] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 3

[22] J. Lawrence, D. B. Goldman, S. Achar, G. M. Blascovich, J. G. Desloge, T. Fortes, E. M. Gomez, S. Häberling, H. Hoppe, A. Huibers, C. Knaus, B. Kuschak, R. Martin-Brualla, H. Nover, A. I. Russell, S. M. Seitz, and K. Tong. Project starline: A high-fidelity telepresence system. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 40(6), 2021. 9

[23] K. Lee, J. Yi, and Y. Lee. Farfetchfusion: Towards fully mobile live 3d telepresence platform. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pp. 1–15, 2023. 8, 9

[24] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. Groot: A real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pp. 1–14, 2020. 8

[25] J. Li, S. Chen, Q. Li, and Z. Liu. Pcvd: A dataset of point cloud video for dynamic human interaction. In *Proceedings of the 16th ACM Multimedia Systems Conference*, MMSys '25, 7 pages, p. 284–290. Association for Computing Machinery, New York, NY, USA, 2025. doi: 10.1145/3712676.3718343 1, 2, 6, 7, 8

[26] Y. Liang, Y. Hu, W. Shao, and Y. Fu. Distilling monocular foundation model for fine-grained depth completion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 22254–22265, 2025. 9

[27] Z. Liang, J. Liu, M. Dasari, and F. Wang. Fumos: Neural compression and progressive refinement for continuous point cloud video streaming. *IEEE Transactions on Visualization and Computer Graphics*, 30(5):2849–2859, 2024. 9

[28] E. Lu, S. Bharadwaj, M. Dasari, C. Smith, S. Seshan, and A. Rowe. Renderfusion: Balancing local and remote rendering for interactive 3d scenes. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 312–321. IEEE, 2023. 9

[29] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2

[30] Microsoft. Azure kinect dk hardware specifications. https://learn.microsoft.com/en-us/previous-versions/azure/kinect-dk/hardware-specification, 2021. 9

[31] O. S. Network. Open storage network. https://www.openstoragenetwork.org/, 2025. 4

[32] OpenCV. Opencv camera calibration. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html, 2025. 3

[33] I. Reimat, E. Alexiou, J. Jansen, I. Viola, S. Subramanyam, and P. Cesar. Cwipc-sxr: Point cloud dynamic human dataset for social xr. In *Proceedings of the 12th ACM Multimedia Systems Conference*, MMSys '21, 7 pages, p. 300–306. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3458305.3478452 1, 2, 6, 8

[34] P. Rim, H. Park, S. Gangopadhyay, Z. Zeng, Y. Chung, and A. Wong. Protodepth: Unsupervised continual depth completion with prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6304–6316, June 2025. 9

[35] J. Shi, M. Zhang, L. Shen, J. Liu, Y. Zhang, L. Pu, and J. Xu. Towards full-scene volumetric video streaming via spatially layered representation and nerf generation. In *Proceedings of the 34th Workshop on Network and Operating System Support for Digital Audio and Video*, NOSSDAV '24, 7 pages, p. 22–28. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3651863.3651879 8

[36] J. Shi, M. Zhang, L. Shen, J. Liu, Y. Zhang, L. Pu, and J. Xu. Implicit representation-based volumetric video streaming for photorealistic full-scene experience. *ACM Trans. Multimedia Comput. Commun. Appl.*, Apr. 2025. Just Accepted. doi: 10.1145/3728472 1, 2, 8

[37] J. Wang and E. Olson. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4193–4198. IEEE, 2016. 3

[38] R. Wang, S. Xu, C. Dai, J. Xiang, Y. Deng, X. Tong, and J. Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5261–5271, 2025. 9

[39] R. Wang, S. Xu, Y. Dong, Y. Deng, J. Xiang, Z. Lv, G. Sun, X. Tong, and J. Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details, 2025. 9

[40] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 9

[41] D. Yin, J. Shi, M. Zhang, Z. Huang, J. Liu, and F. Dong. Fsvfg: Towards immersive full-scene volumetric video streaming with adaptive feature grid. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, 10 pages, p. 11089–11098. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3664647.3680908 9

[42] A. Zhang, C. Wang, B. Han, and F. Qian. {YuZu}:{Neural-Enhanced} volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pp. 137–154, 2022. 8

[43] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*,

2018. 5

[44] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2002. 3

[45] Q.-Y. Zhou, J. Park, and V. Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 3