Gaussian Splatting-based Volumetric Video Compression with Sparse 4D Anchors

Ge Gao[1], **Siyue Teng**[1], **Changqi Wang**[1], **Fan Zhang**[1]
Nantheera Anantrasirichai[1], **Jui Chiu Chiang**[2], **Wen-Hsiao Peng**[3], **David Bull**[1]
[1]Visual Information Lab, University of Bristol, UK
[2]National Chung Cheng University, Taiwan [3]National Yang Ming Chiao Tung University, Taiwan
{ge1.gao, siyue.teng, changqi.wang, fan.zhang, n.anantrasirichai, dave.bull}@bristol.ac.uk
rachel@ccu.edu.tw, wpeng@cs.nycu.edu.tw

Project Page: https://ge1gao.github.io/saga/

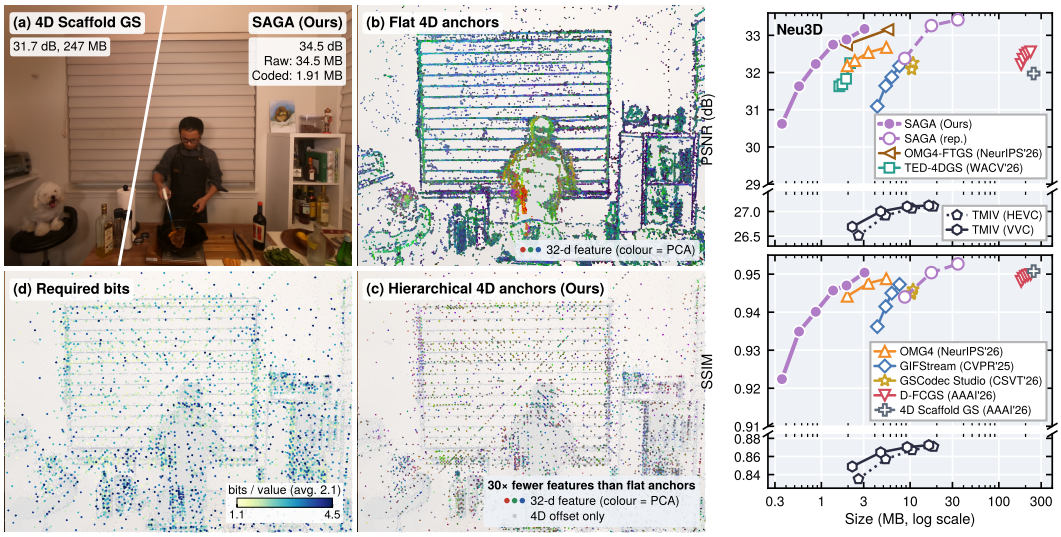


Figure 1: **SAGA: Hierarchical 4D anchors for compact dynamic scenes.** While flat 4D anchors (**b**) store features individually, SAGA (**c**) concentrates features at sparse coarse-level anchors, achieving a 30× reduction in feature-bearing anchors. Our long-context entropy model further exploits dependencies among features, achieving an average coding rate of just 2 bits/symbol (**d**). Together, these designs compress a 300-frame Neu3D scene to 1.91MB—129× smaller than 4D Scaffold-GS (**a**)—while achieving the best rate-distortion trade-off among state-of-the-art baselines (**right**).

Abstract

Immersive video communication requires photorealistic, compact dynamic scene representations that support efficient rendering. 3D Gaussian Splatting (3DGS) is a promising approach, but dynamic 3DGS remains difficult to compress due to its dense primitives and spatiotemporal redundancy. Anchor-based formulations improve compactness through sparse scaffolds that share geometry and appearance across primitives. However, existing designs typically deform a single canonical scaffold and condition each primitive solely on its associated anchor, limiting their ability to model non-local dynamics and disocclusions and exploit inter-anchor correlations. To address these limitations, we propose **SAGA**, a volumetric video codec built on **S**parse **A**nchor-assisted **GA**ussian splatting representations. SAGA

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

represents a dynamic scene using hierarchical sparse anchors in 4D space-time. Coordinate-based INR decoders generate fine anchors and Gaussian primitives from features obtained through kernel interpolation among nearby anchors, enabling parameter sharing across spatiotemporal structures. A fixed-size memory with a novel orthogonality-informed slot-update strategy further captures long-range dependencies among unstructured anchors for entropy coding. Empirically, SAGA achieves PSNR-based BD-rate reductions of **90.88%**, **64.30%** and **65.14%** relative to GIFStream, TED-4DGS, and OMG4, respectively, on the Neu3D testset.

1 Introduction

Immersive visual content is increasingly important for applications such as telepresence, free-viewpoint video, virtual and augmented reality, and digital media production [1, 6]. It provides a stronger sense of presence, richer spatial understanding, and greater viewpoint flexibility than conventional 2D media. However, these benefits introduce substantial challenges. While 2D video mainly records view-specific projected appearance on the imaging plane, free-viewpoint volumetric video requires dynamic scene representations that support rendering from multiple viewpoints, further increasing the demands on storage, transmission, and rendering. Consequently, compact dynamic scene representation has become central to scalable immersive video systems, where fidelity, rendering efficiency, and compressibility must be jointly optimized.

For compact representation and compression of volumetric visual content, prior work spans traditional geometry- and video-based pipelines [40, 39, 1, 6] and neural scene representations [34, 85, 48]. Among neural methods, Neural Radiance Fields (NeRF) have advanced novel-view synthesis [56] and compression [74, 83, 25, 80] by modeling scenes as continuous volumetric radiance fields, but their reliance on volume rendering remains costly for real-time immersive applications. In contrast, 3D Gaussian Splatting (3DGS) [32] uses an explicit, rasterization-friendly representation with efficient training, high visual fidelity, and substantially faster rendering. Consequently, 3DGS compression (e.g., [54, 11]) has become an active research direction for reducing storage costs and enabling practical immersive visual communication.

Following advances in static 3DGS compression [54, 11, 12, 10, 72, 58], GS-based volumetric video coding has attracted increasing attention [67, 24, 41]. Compared with static scenes, dynamic 3DGS must represent changes in geometry, appearance, and visibility over time, introducing additional parameters and spatiotemporal redundancy. Existing methods broadly fall into frame-by-frame approaches [67, 24, 38, 78], which encode temporally correlated Gaussian frames, and canonical-to-deformation approaches [13, 41, 23] that model video- or chunk-wise temporal variations from a shared canonical representation. While effective, frame-wise methods exploit temporal redundancy via predictive or correlated coding, which can complicate random access and accumulate drift under long prediction chains; canonical-to-deformation methods rely on a single reference that can be affected by abrupt motion, disocclusion, or topology changes.

To address these limitations, we propose **SAGA**, a Gaussian Splatting-based volumetric video compression framework using a hierarchical sparse 4D anchor scaffold. SAGA organizes anchors directly in spacetime, allowing features to be shared across spatial and temporal neighborhoods without deforming a single canonical scaffold. Fine anchors and Gaussian primitives are generated through a unified *interpolation-and-decoding* mechanism: neighboring parent-anchor features are aggregated at each target coordinate and fed, together with positional embeddings, into shared coordinate-based implicit neural representation (INR) decoders. This enables parameter sharing at multiple granularities while retaining local flexibility to represent dynamic scene content. A lightweight local hash-based residual branch further refines primitive attributes in normalized anchor coordinates to recover high-frequency details. To complement this local feature sharing, we introduce a long-context entropy model that summarizes previously decoded anchor information in a fixed-size memory bank. Orthogonality-guided slot updates incorporate new context while limiting repeated writes of information aligned with the previous slot state. Our main contributions are as follows:

• We propose a novel **GS-based volumetric video compression framework** with hierarchical sparse 4D anchors. By organizing anchors directly in spacetime, our formulation combines local adaptability to geometry and visibility changes with long-range temporal sharing, enabling bidirectional redundancy reduction and flexible non-canonical modeling of complex dynamics.

• We leverage **coarse-to-fine and inter-anchor correlations** through position-interpolated hierarchical features. These features condition both fine-anchor and Gaussian decoding, allowing them to aggregate neighboring-anchor context rather than relying on isolated anchor features.

• We introduce an efficient **long-context entropy model** for unstructured 4D anchors. It distills decoded spatiotemporal contexts into a compact memory bank whose slots are updated with an orthogonality-guided rule, capturing long-range dependencies under bounded complexity.

Experiments on Neu3D [45], Panoptic Sports [29], and MPEG MIV Common Test Condition (CTC) [6] datasets show that SAGA achieves favorable rate-distortion performance against existing GS-based and MPEG-standard volumetric video codecs.

2 Related Work

Volumetric video compression has traditionally focused on explicit scene data and multi-view signals, such as point clouds and multi-view videos. Representative standardization efforts include V-PCC [63, 26], G-PCC [39], and V3C [1], covering video-based and geometry-based coding of volumetric content. For immersive multi-view video, TMIV [1, 6] provides reference software that exploits inter-view redundancy and packs texture and geometry into video atlases. Video-based volumetric codecs can further leverage mature 2D coding tools and ecosystems, such as HEVC [66] and VVC [7], with performance typically assessed under common test conditions and view-synthesis evaluation protocols. Beyond standardized pipelines, learning-based methods have also been explored for explicit 3D data, especially point clouds, using learned geometry transforms, entropy models, temporal prediction, motion compensation, and residual coding [59, 16, 28, 64]. More recently, coordinate-based and implicit neural representation (INR) codecs have emerged for multi-view and immersive video compression, such as MV-HiNeRV [34], MV-IERV [85], and MV-MGINR [48].

Compression for NeRF-based methods has evolved along a clear methodological progression. Early work improved compactness through end-to-end optimization [4] and more structured radiance-field parameterizations, including sparse grids, explicit voxels [81], tensor decompositions [68, 9], multiresolution hash encodings [57], and plane-based factorization [8, 18]. Later works treat NeRFs more directly as compressible representations, introducing transform coding [60, 37], quantization [42, 31], entropy-constrained coding [37, 43], and learned codec regimes [43, 31, 71, 17, 83, 25]. These studies show that compactness strongly depends on structured field parameterization, but they remain primarily designed for volumetric radiance fields, whose rendering is generally less direct for real-time immersive applications than explicit Gaussian rasterization.

Compression for GS-based methods extends compact static 3DGS representations to dynamic volumetric video, driven by advances in GS compression [54, 11, 12, 10, 67, 24, 41] and multiview datasets [29, 62, 45, 19, 2]. For static 3DGS [32], existing methods reduce redundancy through compact parameterization [79, 14], pruning [61, 22, 35], quantization [70, 58, 15], context modeling [49, 12, 72, 50, 76], and rate-distortion optimization [54, 11, 12, 10]. Feed-forward methods [53, 10, 78] further avoid extensive per-scene overfitting. For dynamic scenes, GS-based codecs can be broadly grouped into frame-wise predictive approaches and temporally shared representations. The former encode timestamp-dependent Gaussian sets using online updates, motion compensation, GoF-based inter-frame coding, residual prediction, or temporal pruning and quantization [67, 20, 24, 78, 21, 52, 27]. The latter model temporal evolution with canonical deformation, feature streams, motion-decoupled trajectories, or structured UV/attribute atlases [41, 82, 84]. Beyond volumetric video, generative GS-based approaches have also been explored for 2D video compression [51, 69].

3 Method

3.1 Preliminaries

Gaussian Splatting represents a 3D scene as learnable anisotropic Gaussian primitives $\{g_j\}_{j=1}^{N}$. Each primitive is parameterized as $g_j = (\boldsymbol{\mu}_j, \boldsymbol{q}_j, \boldsymbol{s}_j, \alpha_j, \boldsymbol{c}_j)$, where $\boldsymbol{\mu}_j \in \mathbb{R}^3$ is the center, $\boldsymbol{q}_j$ denotes the rotation quaternion, $\boldsymbol{s}_j$ is the scale, α_j is the opacity, and $\boldsymbol{c}_j$ encodes view-dependent color/appearance. Its covariance is $\boldsymbol{\Sigma}_j = \boldsymbol{R}_j \operatorname{diag}(\boldsymbol{s}_j)^2 \boldsymbol{R}_j^{\intercal}$. Given a camera, each Gaussian is projected to a 2D elliptical footprint $\mathcal{G}_j(\boldsymbol{p})$. After depth sorting, the color values at pixel $\boldsymbol{p}$ are computed by differentiable front-

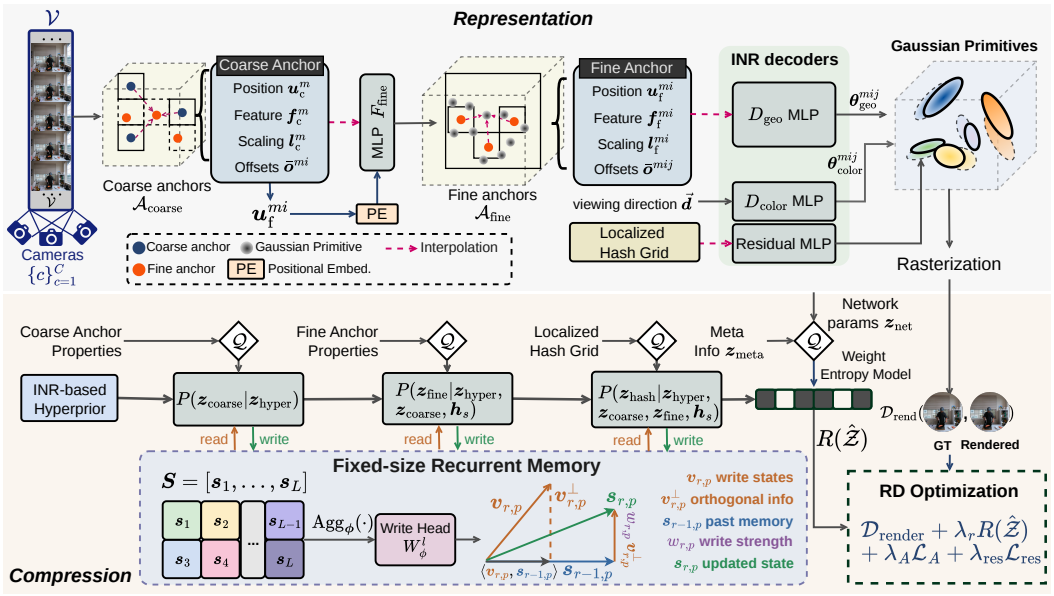


Figure 2: **Overview of SAGA.** In the representation stage, coarse anchors provide shared spatiotemporal context for fine anchors through localized kernel interpolation, while fine anchors generate Gaussian primitives via INR decoders and residual hash refinement. In the compression stage, anchors, hash grids, and neural decoder parameters are quantized and entropy coded with a hierarchical context model, where a recurrent memory bank captures long-range dependencies across sequential coding groups for improved compression efficiency.

to-back alpha compositing as $\boldsymbol{C}(\boldsymbol{p}) = \sum_{j=1}^{N} T_j \alpha_j \mathcal{G}_j(\boldsymbol{p}) \boldsymbol{c}_j$, where $T_j = \prod_{j'=1}^{j-1} (1 - \alpha_{j'} \mathcal{G}_{j'}(\boldsymbol{p}))$ is the accumulated transmittance. This explicit differentiable formulation enables efficient rendering and high-quality novel view synthesis.

ScaffoldGS [54] organizes 3D Gaussians with a sparse set of anchor points. Each anchor i is associated with a position $\boldsymbol{x}_i$, an anchor feature $\boldsymbol{f}_i$, a learnable scaling factor $\boldsymbol{l}_i$, and K learnable local offsets $\boldsymbol{O}_i = \{\boldsymbol{o}_{i,k}\}_{k=1}^{K}$. For a visible anchor, ScaffoldGS spawns K neural Gaussians whose centers are given by:

$$\boldsymbol{\mu}_{i,k} = \boldsymbol{x}_i + \boldsymbol{o}_{i,k} \odot \boldsymbol{l}_i \tag{1}$$

The remaining Gaussian attributes, including opacity, color, rotation, and scale, are not independently stored for each spawned Gaussian. Instead, they are decoded on the fly from a view-adaptive anchor feature together with the camera-anchor distance δ_i and viewing direction $\vec{\boldsymbol{d}}_i$:

$$\{\alpha_{i,k}\}_{k=1}^{K} = F_\alpha(\hat{\boldsymbol{f}}_i, \delta_i, \vec{\boldsymbol{d}}_i), \quad \{\boldsymbol{c}_{i,k}\}_{k=1}^{K} = F_c(\hat{\boldsymbol{f}}_i, \delta_i, \vec{\boldsymbol{d}}_i),$$

with $\{\boldsymbol{q}_{i,k}\}_{k=1}^{K}$ and $\{\boldsymbol{s}_{i,k}\}_{k=1}^{K}$ predicted analogously by F_q and F_s. Thus, ScaffoldGS can be viewed as an anchor-conditioned neural Gaussian generator, where sparse anchors provide local context and offsets, while lightweight MLPs instantiate view-adaptive Gaussian attributes.

3.2 Overview

The proposed SAGA framework is illustrated by Figure 2, which consists of two major stages: representation and compression, which we describe in detail below:

Representation. To exploit correlations among unstructured 4D anchors, we leverage a unified *interpolation-and-decoding* mechanism across the hierarchy. For each fine anchor, the neighboring coarse anchors of its parent are aggregated via kernel interpolation, as detailed in subsection 3.3, producing a coordinate-adaptive latent feature $\tilde{\boldsymbol{f}}_c(\boldsymbol{u}_f^{mi})$, which includes a positional embedding of $\boldsymbol{u}_f^{mi}$. The fine-anchor offset $\bar{\boldsymbol{o}}^{mi}$ is part of the encoded scaffold and is used, after positional embedding, as the local coordinate input to the shared INR decoder. Specifically, F_{fine} predicts the

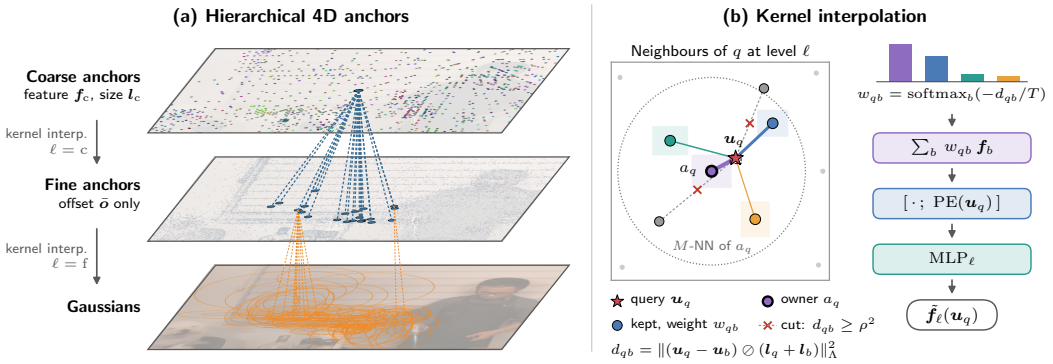


Figure 3: **Hierarchical 4D anchors and kernel interpolation.** (a) Sparse coarse anchors store features, fine anchors store only offsets, and features reach fine anchors and Gaussians by kernel interpolation; (b) a query blends its neighbours' features, weighted by support-normalized distance.

remaining fine-anchor attributes as

$$(\boldsymbol{f}_{\mathrm{f}}^{mi}, \boldsymbol{l}_{\mathrm{f}}^{mi}) = F_{\mathrm{fine}}\left(\tilde{\boldsymbol{f}}_{\mathrm{c}}(\boldsymbol{u}_{\mathrm{f}}^{mi}), \mathrm{PE}(\bar{\boldsymbol{o}}^{mi})\right). \tag{2}$$

Here $\boldsymbol{f}_{\mathrm{f}}^{mi}$ and $\boldsymbol{l}_{\mathrm{f}}^{mi}$ are the features and the support of the fine anchor, respectively. $\mathrm{PE}(\cdot)$ denotes positional embedding [65]. Gaussian primitives are generated from the fine-anchor scaffold in the same local-coordinate manner. Each fine anchor carries a set of primitive offsets $\{\bar{\boldsymbol{o}}^{mij}\}_{j=1}^{N_{mi}}$ in its local coordinate system, obtained as $\bar{\boldsymbol{o}}^{mij} = F_{\mathrm{off}}(\boldsymbol{f}_{\mathrm{f}}^{mi})_j + \boldsymbol{r}^{mij}$, where the shared predictor F_{off} provides a layout template and only the residuals $\boldsymbol{r}^{mij}$ are transmitted as part of the scaffold. For each primitive, we recover its 4D coordinate $\boldsymbol{u}_{\mathrm{g}}^{mij}$ from $\bar{\boldsymbol{o}}^{mij}$ and the parent fine anchor, interpolate neighboring fine anchors to obtain $\tilde{\boldsymbol{f}}_{\mathrm{f}}(\boldsymbol{u}_{\mathrm{g}}^{mij})$, again fused with $\mathrm{PE}(\boldsymbol{u}_{\mathrm{g}}^{mij})$, and decode its attributes as

$$\boldsymbol{\theta}_{\mathrm{geo}}^{mij} = F_{\mathrm{geo}}\left(\tilde{\boldsymbol{f}}_{\mathrm{f}}(\boldsymbol{u}_{\mathrm{g}}^{mij}), \mathrm{PE}(\bar{\boldsymbol{o}}^{mij})\right), \qquad \boldsymbol{c}_{c,s}^{mij} = F_{\mathrm{color}}\left(\tilde{\boldsymbol{f}}_{\mathrm{f}}(\boldsymbol{u}_{\mathrm{g}}^{mij}), \mathrm{PE}(\bar{\boldsymbol{o}}^{mij}), \vec{\boldsymbol{d}}_c^{mij}, \tau_s - \tau^{mij}\right). \tag{3}$$

Here $\boldsymbol{\theta}_{\mathrm{geo}}^{mij}$ includes opacity, spatial scaling, base rotation, temporal-opacity parameters, and polynomial motion/rotation coefficients following STG [46]. These coefficients are evaluated at timestamp τ_s to obtain primitive position and orientation, while F_{color} predicts view- and time-dependent color. Each primitive further carries a learnable binary mask m^{mij} [12], optimized with a straight-through estimator, which gates its opacity during training and to remove/exclude invalid Gaussians. The retained primitives are rasterized to produce $\tilde{I}_{c,s}$.

Compression. Parameters are quantized with learned quantization step sizes, using soft-rounding [33] during training, followed by a short quantization-noise stage. With a slight abuse of notation, we denote the tuple of quantized parameters to be entropy coded as $\hat{\mathcal{Z}} = (\hat{\boldsymbol{z}}_{\mathrm{meta}}, \hat{\boldsymbol{z}}_{\mathrm{coarse}}, \hat{\boldsymbol{z}}_{\mathrm{fine}}, \hat{\boldsymbol{z}}_{\mathrm{prim}}, \hat{\boldsymbol{z}}_{\mathrm{mask}}, \hat{\boldsymbol{z}}_{\mathrm{net}})$, including meta information, coarse/fine anchor symbols, primitive offset residuals, primitive masks, and network parameters, with the latter also covering the INR-based hyperprior [23]. We encode $\hat{\boldsymbol{z}}_{\mathrm{meta}}$ independently and compress $\hat{\boldsymbol{z}}_{\mathrm{net}}$, which includes the decoders and the entropy model itself; the remaining symbols are entropy modeled hierarchically, as shown in Figure 2, conditioned on the hyperprior, previously decoded neighbors, and a memory bank. Within the coarse level, and likewise for the primitive offsets, symbols are partitioned into K_{c} parity-based coding groups according to the integer 4D cell coordinates of their anchors. For the n-th anchor with cell coordinate $\boldsymbol{v}_n^{\ell} \in \mathbb{Z}^4$, the group index is

$$\gamma(\boldsymbol{v}_n^{\ell}) = \boldsymbol{\omega}^{\intercal}(\boldsymbol{v}_n^{\ell} \bmod 2), \qquad \boldsymbol{\omega} = (1, 2, 4, 8)^{\intercal}, \tag{4}$$

so that the k-th coding group is $\mathcal{G}_{\ell,k} = \{n \mid \gamma(\boldsymbol{v}_n^{\ell}) = k\}$ for $k = 0, \ldots, K_{\mathrm{c}} - 1$, with $K_{\mathrm{c}} = 16$. The groups are decoded sequentially over k, while all symbols within the same group are decoded in parallel. Fine-anchor offsets are decoded in a single step conditioned on their parent coarse anchors. Decoding the full scaffold therefore requires $2K_{\mathrm{c}} + 1$ sequential steps.

3.3 Hierarchical 4D Anchor

Initialization and Optimization. We initialize the hierarchy from a unified 4D candidate pool. The sequence is split into groups of pictures (GOPs), and a short spatio-temporal Gaussian (STG) [46] warm-up is fitted to each GOP as a proposal stage. For every provisional Gaussian q, one gradient pass over the training views accumulates its screen-space gradient magnitude S_q and a temporally normalized response D_q that favors short-lived content. Gaussians whose S_q or D_q exceeds the corresponding upper quantile, and whose opacity is non-negligible, become candidates with importance

$$w_q = \min\left\{\max\left(\frac{S_q}{Q_S}, \frac{D_q}{Q_D}\right)\frac{\bar{d}}{d_q},\ w_{\max}\right\}, \tag{5}$$

where Q_S and Q_D are the corresponding quantile thresholds, d_q is the camera distance of candidate q, and $\bar{d}$ is a reference camera distance used for normalization. A candidate is treated as static when it barely moves and remains visible for at least half a GOP. Static candidates are placed at the middle timestamp and de-duplicated across GOPs, while dynamic candidates are mapped to sequence time.

The resulting pool is $\mathcal{P} = \{((\boldsymbol{x}_q, \tau_q), \boldsymbol{h}_q, w_q)\}_q$, where $(\boldsymbol{x}_q, \tau_q)$, $\boldsymbol{h}_q$, and w_q denote the 4D location, color, and importance, respectively. We build a greedy dyadic 4D tree using the descriptor

$$\boldsymbol{\phi}_q = [\boldsymbol{x}_q,\ \lambda_\tau \tau_q,\ \lambda_h \boldsymbol{h}_q], \tag{6}$$

so subdivision reflects spatial, temporal, and appearance variation. The leaf with the largest importance-weighted descriptor scatter (sum of squared deviations) is split next, until the coarse-anchor budget or maximum depth is reached. A cell is split in space only, unless the weighted temporal spread within it exceeds a threshold, in which case it is also halved in time.

Leaf cells automatically become coarse anchors. For leaf m, $\boldsymbol{u}_{\mathrm{c}}^m$ is the cell center, $\boldsymbol{l}_{\mathrm{c}}^m$ is the cell half-size, refined during training by a learned log-scale correction, and $\boldsymbol{f}_{\mathrm{c}}^m$ is a learnable feature initialized from the weighted mean colour of its candidates. Fine-anchor capacity is allocated according to

$$A_m = \left(\textstyle\sum_{q\in\mathcal{P}_m} w_q\right)^{\alpha} (H_m + \epsilon)^{\beta}, \tag{7}$$

where $\mathcal{P}_m$ is the candidate set in cell m and H_m is its importance-weighted descriptor variance. The allocation scores are converted into integer counts N_m subject to $\sum_m N_m = N_{\mathrm{budget}}, N_{\min} \le N_m \le N_{\max}$. Given N_m, weighted k-means++ seeding selects representatives, which are stored as normalized offsets, i.e., $\bar{\boldsymbol{o}}^{mi} = \left(\boldsymbol{u}^{mi} - \boldsymbol{u}_{\mathrm{c}}^m\right) \oslash \boldsymbol{l}_{\mathrm{c}}^m$, where $\oslash$ denotes element-wise division.

Geometry Coding and Decoding Order. We use c and f to denote the coarse and fine anchor levels, respectively. Each coarse center is quantized on a $2^{B_{\mathrm{c}}}$ grid per dimension and mapped to a 4D Morton key κ_m^{c} by interleaving the coordinate bits of (x, y, z, τ). The sorted keys are stored with Elias–Fano coding, followed by the entropy-coded cell depths and fine-anchor counts N_m, so the tree layout is encoded losslessly. Fine anchors are stored as offsets $\bar{\boldsymbol{o}}^{mi}$ from their parent, quantized and entropy-coded with the parent coarse anchor as context.

Attributes are decoded level by level, so each level is conditioned on the preceding one: coarse features and support deltas, then fine offsets, then Gaussian masks and offsets. Within the coarse and Gaussian levels, symbols are split into 16 groups by the parity of their 4D cell index (for a Gaussian, the local cell of its fine anchor). Group k is decoded in parallel, conditioned on its decoded neighbours in groups smaller than k, which bounds the sequential depth to 16 steps per level.

Kernel Interpolation. Let $\ell \in \{\mathrm{c}, \mathrm{f}\}$ denote the source level, used to decode fine anchors from coarse anchors ($\ell = \mathrm{c}$) and Gaussian primitives from fine anchors ($\ell = \mathrm{f}$). Each source anchor b has position $\boldsymbol{u}_b$, support $\boldsymbol{l}_b$, and feature $\boldsymbol{f}_b$. A query at $\boldsymbol{u}_q$ belongs to a source anchor a_q (its parent or owner) and has support $\boldsymbol{l}_q$, set to $\frac{1}{4}\boldsymbol{l}_{a_q}$ for fine anchors and $\boldsymbol{l}_{a_q}$ for Gaussians. Its candidates are the M nearest 4D neighbours of a_q by center distance, which the decoder recomputes for free and which also provide the entropy-coding context of the coarse and Gaussian levels. Among them, the K closest with $d_{qb} < \rho^2$ form $\mathcal{N}(q)$ (always including a_q), and their features are blended , as illustrated in Figure 3 (b):

$$\tilde{\boldsymbol{f}}_\ell(\boldsymbol{u}_q) = \mathrm{MLP}_\ell\left(\Big[\sum_{b\in\mathcal{N}(q)} \mathrm{softmax}_b\left(-d_{qb}/T\right)\boldsymbol{f}_b\,;\,\mathrm{PE}(\boldsymbol{u}_q)\Big]\right), \quad d_{qb} = \left\|(\boldsymbol{u}_q - \boldsymbol{u}_b) \oslash (\boldsymbol{l}_q + \boldsymbol{l}_b)\right\|_{\boldsymbol{\Lambda}}^2, \tag{8}$$

where $[\cdot\,;\cdot]$ denotes concatenation, PE is a sinusoidal encoding, MLP_ℓ is a two-hidden-layer ReLU MLP per level, $\oslash$ is element-wise division, and $\boldsymbol{\Lambda} = \mathrm{diag}(1, 1, 1, \lambda_t)$ weights the temporal term.

3.4 Hierarchical Context Modeling

Our entropy model augments the local hierarchical context with a fixed-size memory bank to capture long-range dependencies among sparse 4D anchors. Let r index the sequential decoding steps induced by the hierarchy level and modulo group; for example, one step may decode the symbol blocks associated with $\mathcal{G}_{\ell,k}$, where $\ell \in \{\mathrm{c}, \mathrm{f}\}$ and k is the modulo-group index. We denote the blocks decoded at step r as $\mathcal{B}_r$. All blocks in $\mathcal{B}_r$ are decoded in parallel and read from the same previous memory state $\boldsymbol{S}_{r-1} = [\boldsymbol{s}_{r-1,1}, \ldots, \boldsymbol{s}_{r-1,L}]$, where L is the number of memory slots.

For each symbol block $g \in \mathcal{B}_r$, we form a memory query token $\boldsymbol{q}_g$ from its known side information and read the memory as $\boldsymbol{h}_g^{\mathrm{mem}} = \mathrm{Read}_\phi(\boldsymbol{q}_g, \boldsymbol{S}_{r-1})$. The entropy parameters are then predicted by $(\boldsymbol{\mu}_g, \boldsymbol{\sigma}_g) = E_\phi(\boldsymbol{e}_g^{\mathrm{hp}}, \boldsymbol{h}_g^{\mathrm{parent}}, \boldsymbol{h}_g^{\mathrm{prev}}, \boldsymbol{h}_g^{\mathrm{mem}})$, where $\boldsymbol{e}_g^{\mathrm{hp}}$ is the hyperprior embedding, $\boldsymbol{h}_g^{\mathrm{parent}}$ is aggregated from decoded coarser-level symbols, and $\boldsymbol{h}_g^{\mathrm{prev}}$ uses only symbols from earlier modulo groups within the same coding stage. No symbol from the current batch $\mathcal{B}_r$ is used to predict another symbol in the same batch, preserving parallel entropy decoding.

After all blocks in $\mathcal{B}_r$ are decoded, their quantized values and side information are converted into decoded tokens $\{\tilde{\boldsymbol{z}}_g\}_{g\in\mathcal{B}_r}$ and aggregated into a batch write token $\boldsymbol{a}_r = \mathrm{Agg}_\phi(\{\tilde{\boldsymbol{z}}_g\}_{g\in\mathcal{B}_r})$. The memory bank contains L slots, and $p \in \{1, \ldots, L\}$ indexes an individual slot. For each slot p, a slot-specific candidate write vector and gate are predicted from the same batch write token: $(\boldsymbol{v}_{r,p}, w_{r,p}) = W_\phi^p(\boldsymbol{a}_r)$. To avoid repeatedly writing information already represented by the previous slot, we remove from the candidate write vector the component parallel to that slot:

$$\boldsymbol{v}_{r,p}^{\perp} = \boldsymbol{v}_{r,p} - \frac{\langle \boldsymbol{v}_{r,p}, \boldsymbol{s}_{r-1,p} \rangle}{\|\boldsymbol{s}_{r-1,p}\|_2^2 + \epsilon} \boldsymbol{s}_{r-1,p}. \tag{9}$$

Each memory slot is then updated from its own previous value by combining the previous slot with the gated orthogonal write:

$$\boldsymbol{s}_{r,p} = \mathrm{Norm}\left(\boldsymbol{s}_{r-1,p} + w_{r,p}\boldsymbol{v}_{r,p}^{\perp}\right), \qquad p = 1, \ldots, L. \tag{10}$$

After all L slots have been updated, the new memory state is formed as $\boldsymbol{S}_r = [\boldsymbol{s}_{r,1}, \ldots, \boldsymbol{s}_{r,L}]$. Here $\boldsymbol{s}_{r-1,p}$ is the previous value of slot p, $\boldsymbol{v}_{r,p}^{\perp}$ is the orthogonalized write content for that slot, $w_{r,p}$ controls the slot-specific write strength, and $\boldsymbol{s}_{r,p}$ denotes the updated slot. All slot updates are computed with respect to the previous memory state $\boldsymbol{S}_{r-1}$ and the batch-level write token $\boldsymbol{a}_r$.

3.5 Rate-Distortion Optimization

After the proposal warm-up and hierarchy construction, the provisional flat anchors are discarded, and only the hierarchical 4D anchor scaffold is optimized for compression. We train SAGA with a rate-distortion objective over the quantized transmitted symbols $\hat{\mathcal{Z}}$:

$$\mathcal{L} = \mathcal{D}_{\mathrm{render}} + \lambda_r R(\hat{\mathcal{Z}}) + \lambda_A \mathcal{L}_A + \lambda_{\mathrm{mask}} \mathcal{L}_{\mathrm{mask}}. \tag{11}$$

Here $R(\hat{\mathcal{Z}})$ is the entropy-estimated bitrate under the hierarchical coding order, and λ_r, λ_A and λ_{mask} control the relative strengths of the bitrate penalty, anchor regularization, and mask regularization terms, respectively. Quantization and the rate term are enabled after an initial reconstruction stage, with λ_r ramped up linearly. The rendering loss follows common 3DGS practice, with $\mathcal{D}_{\mathrm{render}} = (1-\eta)\|\tilde{I}_{c,s} - I_{c,s}\|_1 + \eta(1 - \mathrm{SSIM}(\tilde{I}_{c,s}, I_{c,s}))$, where $\tilde{I}_{c,s}$ and $I_{c,s}$ are the rendered and ground-truth frames; during training, $\tilde{I}_{c,s}$ passes through a per-camera affine color correction that is the identity at test time. The anchor regularization constrains local offsets to remain within their parent-anchor coordinate ranges. We define $\mathcal{O}_A = \{\bar{\boldsymbol{o}}^{mi}\} \cup \{\bar{\boldsymbol{o}}^{mij}\}$ for the coarse-to-fine and fine-to-primitive local offsets, and use $\mathcal{L}_A = |\mathcal{O}_A|^{-1} \sum_{\bar{\boldsymbol{o}}\in\mathcal{O}_A} \|\,\mathrm{ReLU}(|\bar{\boldsymbol{o}}| - \rho_A)\|_2^2$, where ρ_A is the allowed normalized boundary. This prevents the hierarchy from degenerating into unconstrained flat anchors.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct experiments on three datasets: the widely used Neu3D [45], Panoptic Sports [30], and the multi-view texture-and-depth video sequences defined in the MPEG Immersive Video

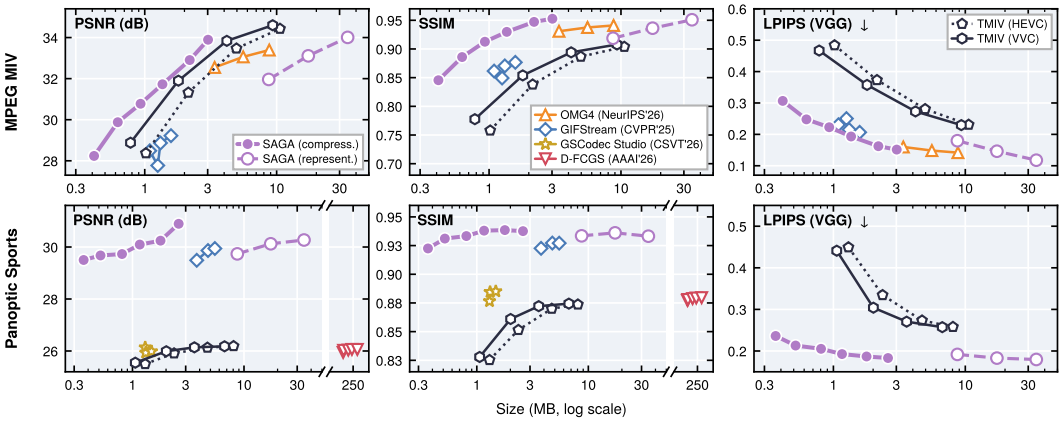


Figure 4: **RD comparison** on Panoptic Sports and MPEG MIV in terms of PSNR, SSIM and LPIPS.

Table 1: **Novel View Synthesis.** We report PSNR, SSIM, LPIPS [77], training time in hours, decoding FPS, and model size on Neu3D and Panoptic Sports. Missing entries are denoted by "–" when results or implementations are unavailable. For a fair comparison, we report the *raw model size* of the proposed SAGA, instead of the size after quantization and entropy coding.

	Neu3D [45]						Panoptic Sports [30]					
Method	PSNR↑	SSIM↑	LPIPS (ALEX)↓	Train↓	FPS↑	Size (MB)↓	PSNR↑	SSIM↑	LPIPS (VGG)↓	Train↓	FPS↑	Size (MB)↓
4DGS [73] (CVPR'24)	31.57	0.947	0.057	120	228	202	28.68	0.911	0.157	–	–	–
E-D3DGS [3] (ECCV'24)	31.42	0.945	0.037	–	–	137	25.61	0.896	0.172	–	50	297.9
STG [46] (CVPR'24)	32.05	0.946	0.050	46	140	200	25.09	0.900	0.181	25	265	181
MEGA [79] (ICCV'25)	31.49	0.956	0.057	–	–	25	–	–	–	–	–	–
4DScaffold [14] (AAAI'26)	31.42	0.945	0.020	145	135	236	27.94	0.925	0.217	121	177	193.7
SAGA (ours) [*small*]	31.53	0.931	0.039	34	247	8.71	29.74	0.935	0.192	31	117	8.71
SAGA (ours) [*medium*]	32.44	0.945	0.021	58	202	17.37	30.12	0.936	0.182	79	112	17.37
SAGA (ours) [*large*]	32.81	0.951	0.019	103	165	32.54	30.43	0.937	0.179	98	110	31.54

Table 2: **4D Video Compression.** Comparison on Neu3D, Panoptic Sports, and MPEG MIV, evaluated in terms of PSNR, SSIM, LPIPS (VGG), and decoding FPS. Each BD-rate value is computed using the corresponding benchmark codec as the anchor. We only report baselines for which BD-rate can be reliably computed (i.e., with sufficient overlaps in rate-distortion range).

	Neu3D [45]				Panoptic Sports [30]				MPEG MIV [6]			
Method	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑
Conventional Volumetric Codec												
TMIV-24.0 (HEVC, main) [6]	–	–	–	0.05	–	–	–	0.07	-45.89%	-82.97%	-85.05%	0.06
TMIV-24.0 (VVEnc, main) [6]	–	–	–	0.03	–	–	–	0.05	-18.41%	-75.24%	-72.27%	0.05
GS-based Methods												
4DGC [24] (CVPR'25)	-62.39%	-41.59%	-61.35%	110	–	–	–	–	-32.38%	-36.13%	-42.61%	115
GIFStream [41] (CVPR'25)	-90.88%	-81.72%	-69.55%	91	-84.87%	-87.06%	-76.95%	99	-67.09%	-67.10%	-45.87%	67
TED-4DGS [23] (WACV'26)	-64.30%	–	–	–	–	–	–	–	–	–	–	–
IC-4DGS [75] (ICIP'26)	-22.62%	–	–	–	–	–	–	–	–	–	–	–
D-FCGS [78] (AAAI'26)	-99.31%	-99.08%	-92.05%	4.9	–	–	–	6.7	–	–	–	–
OMG4 [36] (NeurIPS'26)	-65.14%	-17.01%	-24.28%	245	–	–	–	–	–	–	–	–
SAGA (ours)	**0.00%**	**0.00%**	**0.00%**	85	**0.00%**	**0.00%**	**0.00%**	–	**0.00%**	**0.00%**	**0.00%**	97

(MIV) Common Test Conditions [6]. Neu3D consists of 6 indoor dynamic multi-view sequences captured by 18-21 cameras at a resolution of 2704×2028. Following prior works [41, 78], we represent and compress sequences in Neu3D at *half resolution* for the first 300 frames [41]. Panoptic Sports [30] contains six dynamic sports sequences at 640×360 resolution, each recorded by 31 cameras; following [41], we report results on the *basketball* and *boxes* sequences. For each sequence, cameras 0, 10, 15, and 30 are used for testing, while the remaining cameras are utilized for training according to the protocol in [55]. The MPEG TMIV CTC dataset comprises 18 scenes captured by 9-16 cameras at 1080p resolution, from which we select the first 65 frames of mandatory sequences, as specified by the MPEG MIV CTC (see details in **Appendix**C.2), for benchmarking.

Baselines. For novel view synthesis, we compare SAGA against state-of-the-art compact GS-based methods, including 4DGS [73], STG [46], E-D3DGS [3], MEGA [79], 4DScaffold [14]. For volumetric video compression, we evaluate SAGA against a broad range of approaches, including

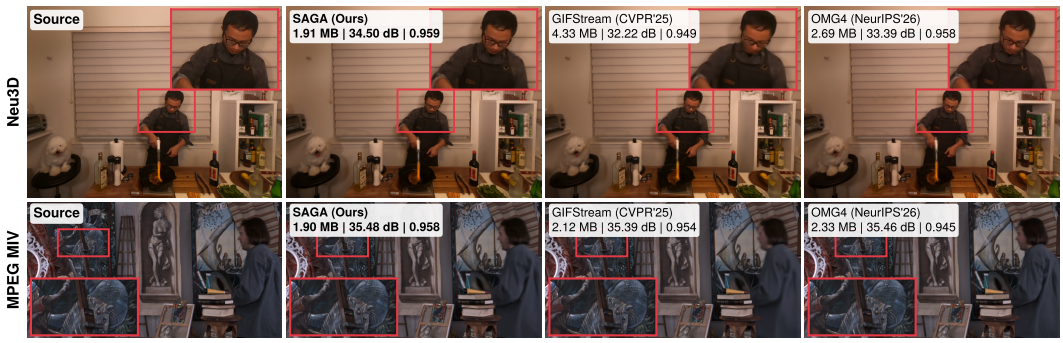


Figure 5: **Exemplary qualitative comparisons** on Neu3D (**top**) and MPEG MIV (**bottom**).

the conventional MPEG immersive video test model TMIV-24.0 [6] in the *MIV Main* configuration, using HM-18.0 and VVEnc-1.12.0 for texture and geometry atlas coding. We use the provided source-view depths for MIV-CTC and estimate depths for Neu3D and Panoptic Sports using Depth-Anything-3 [47], aligning the depth scales with the camera calibration (see details in **Appendix** C.3). We also compare against GS-based compression methods, including 4DGC [24], GIFStream [41], TED-4DGS [23], IC-4DGS [75], D-FCGS [78], GSCodec Studio [44], and OMG4 [36]. It is noted that we report BD-rate results only for methods whose rate-distortion ranges overlap sufficiently with ours to support reliable computation, which, as a consequence, excludes GSCodec Studio and the FTGS variant of OMG4 from Table 2.

Metrics. We use PSNR, SSIM, and LPIPS [77] to measure the reconstruction quality of the synthesized multiviews. For 4D volumetric compression, we report the Bjøntegaard Delta rate (BD-rate) [5] to quantify the relative compression efficiency between different codecs.

4.2 Experimental Results

Novel View Synthesis. As summarized in Table 1, we first evaluate SAGA as a compact dynamic GS representation before quantization and entropy coding. Across Neu3D and Panoptic Sports, SAGA provides a favorable fidelity-size trade-off, with its small variant achieving the smallest reported raw model size among compared methods. On Neu3D, the medium variant already surpasses all evaluated baselines in PSNR with a 17.37 MB model, while the large variant further achieves the best PSNR and LPIPS at 32.81 dB and 0.019, respectively. On Panoptic Sports, even the small variant outperforms all evaluated baselines in PSNR and SSIM, using only 8.7 MB. Increasing model capacity further improves reconstruction quality, although the LPIPS remains higher than that of 4DGS and E-D3DGS. These results show that the proposed spacetime anchor hierarchy and inter-anchor decoding provide a compact representation with competitive reconstruction quality before entropy coding; additional gains from quantization and entropy modeling are evaluated in Table 2.

Volumetric Video Compression. As summarized in Table 2, SAGA achieves consistent rate-distortion improvements over the evaluated GS-based volumetric codecs across the available comparisons. Compared with GIFStream [41], SAGA obtains large BD-rate reductions across all three datasets, with at least 67.09%, 67.10%, and 45.87% savings measured by PSNR, SSIM, and LPIPS, respectively. The gains also extend to recent baselines on Neu3D, with PSNR BD-rate savings exceeding 60% over both TED-4DGS and OMG4, indicating that the compression advantage is not limited to a single baseline. Beyond GS-based methods, SAGA also outperforms both TMIV configurations on MPEG MIV. Even against the VVEnc configuration, it achieves a PSNR BD-rate reduction of 18.41%, with larger savings of over 70% under SSIM and LPIPS. Together, these results demonstrate improved compression efficiency under both distortion-based and perceptual quality metrics. The corresponding rate-distortion curves and qualitative comparisons are provided in Figure 4 and Figure 5, with the latter further illustrating that SAGA preserves finer texture details at lower bitrates than GIFStream and OMG4.

Complexity and Rate Distribution. We measure training cost in GPU-hours and decoding speed in FPS on a single NVIDIA RTX 3090 GPU, and report the component-wise bitrate and runtime breakdown of SAGA in Table 3 (**a**). As summarized in Table 1 and Table 2, the compression gains of SAGA are achieved while retaining real-time decoding, reaching 85 FPS on Neu3D and 97 FPS

Table 3: **Ablation, memory-slot analysis, and bitstream breakdown on Neu3D.** BD-rate is computed in terms of PSNR using SAGA as the anchor.

(a) Bitstream

Component	Before EC MB (%)	After EC MB (%)
Coarse anc.	3.66 (10.6)	0.259 (13.7)
Fine anc.	2.75 (8.0)	0.672 (35.4)
Gauss. offsets	27.47 (79.5)	0.633 (33.4)
Gauss. mask	–	0.116 (6.1)
MLPs	0.67 (1.9)	0.086 (4.5)
Entropy	–	0.021 (1.1)
Side info	–	0.111 (5.8)
Total	34.54	1.897

(b) Component ablation

Ver.	C/F	Interp.	Res.	MLP-C	Upd.	BD-rate↓
V1	✗	✓	✓	✓	Orth.	13.04%
V2	✓	✗	✓	✓	Orth.	11.24%
V3	✓	✓	✗	✓	Orth.	1.03%
V4	✓	✓	✓	✗	Orth.	7.22%
V5	✓	✓	✓	✓	Non-orth.	8.17%
SAGA	✓	✓	✓	✓	Orth.	0.0%

C/F: coarse-to-fine scaffold; Interp.: inter-anchor interpolation; Res.: residual hash; MLP-C: MLP compression; Upd.: memory update. Orth.: orthogonality-guided; Non-orth.: gatedresidual without orthogonal projection.

(c) Memory slots

# Slots	BD-rate↓	FPS↑
32	+8.87%	93.1
64 (default)	0.0%	85.3
128	-10.27%	68.9
256	-15.66%	50.3
512	-16.19%	36.6

on MPEG MIV. Its decoding throughput is comparable to GIFStream on Neu3D and higher on MPEG MIV, although faster methods such as OMG4 offer a different trade-off between compression efficiency and decoding speed. Training remains a notable cost: the small variant requires less training time than 4DScaffold on both datasets, but more than STG. In terms of rate distribution (Table 3 (a)), quantization and entropy coding reduce the anchor scaffold (coarse anchors, fine anchors and Gaussian offsets) from 33.87 MB to 1.56 MB, a 95.4% reduction, with the largest gain on the Gaussian offsets ($43\times$). Fine anchors and Gaussian offsets nevertheless still account for about 69% of the coded bitstream, indicating that scaffold coding remains the main contributor to the total bitrate.

4.3 Ablation Study

To investigate different design components in SAGA, we created the following variants: V1 removes the coarse-to-fine scaffold and uses flat anchors; V2 disables inter-anchor feature interpolation and decodes finer representations only from its own anchor; V3 removes the residual hash grid; V4 disables MLP compression; and V5 replaces the proposed orthogonality-guided memory update with a non-orthogonal gated residual update. The full model is used as the anchor for BD-rate computation. As shown in Table 3 **(b)**, each ablated variant incurs BD-rate loss, demonstrating the effectiveness of the corresponding component. Removing the hierarchy or inter-anchor interpolation substantially degrades RD performance, supporting the importance of both components, while the 8.17% loss of V5 further confirms the contribution of the orthogonalized memory update.

Impact of Memory Slot Size. We also ablate the impact of memory slot size on compression performance and decoding speed, as shown in Table 3 **(c)**. It can be observed that naively increasing the memory slot size offers diminishing returns in compression efficiency; this indicates that the long-context spatiotemporal dependency is already well captured with fewer memory slots, i.e., no more than 256. We hence choose 64 slots as the default for real-time decoding, while larger memories provide better compression at a substantial throughput cost.

5 Conclusion

In this paper, we presented **SAGA**, a GS-based volumetric video compression framework built on hierarchical sparse 4D anchor scaffolds, which exploit inter-anchor redundancy and model local dynamics at variable granularity. Sparse coarse anchors decode fine anchors, which instantiate Gaussian primitives via learned offsets, and learnable masks prune redundant primitives. A long-context entropy model further summarizes decoded contexts into bounded memory for efficient entropy-parameter prediction. Experiments on Neu3D, Panoptic Sports, and MPEG MIV CTC benchmarks demonstrate that SAGA outperforms existing GS-based codecs in compression efficiency, with average BD-rate reductions (measured by PSNR) of 64.30%, 65.14%, and 90.88%, over TED-4DGS, OMG4, and GIFStream on the Neu3D testset, respectively.

Limitations and Future Work. SAGA still requires per-scene optimization, which may incur notable encoding time for long or complex sequences, and fast motion, topology changes, and sparse camera coverage may also challenge reconstruction. Future work will explore faster encoding, adaptive anchor allocation, and standardized bitstreams for scalability, random access, and deployment.

Acknowledgments and Disclosure of Funding

This work was funded by the UKRI MyWorld Strength in Places Programme (SIPF00006/1) and the Royal Society International Exchanges programme (IEC\R3\253264). We thank Mr He-Bi Yang for providing the RD numbers for TED-4DGS and IC-4DGS. We thank the University of Bristol Advanced Computing Research Centre and Isambard-AI for providing computational resources. Isambard-AI is funded by the UK Government via UKRI and STFC [ST/AIRR/I-A-I/1023].

References

[1] V. Allié, I. Bouazizi, N. Duval, E. F. d'Arcier, T. Filoche, P. Fontaine, B. Kroon, T. Stockhammer, D. Silhavy, and J. J. Gimenez. A Comprehensive Suite of Open-Source Reference Tools for MPEG-I Applications: MPEG-I Scene Description and V3C Immersive Platform. In *Proceedings of the 4th Mile-High Video Conference*, pages 101–102, 2025.

[2] A. Azzarelli, G. Gao, H. M. Kwan, F. Zhang, N. Anantrasirichai, O. Moolan-Feroze, and D. Bull. ViVo: A dataset for volumetric video reconstruction and compression. *IEEE TCSVT*, 2026.

[3] J. Bae, S. Kim, Y. Yun, H. Lee, G. Bang, and Y. Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In *ECCV*, pages 321–335. Springer, 2024.

[4] T. Bird, J. Ballé, S. Singh, and P. A. Chou. 3d scene compression through entropy penalized neural representation functions. In *PCS*, pages 1–5. IEEE, 2021.

[5] G. Bjøntegaard. Calculation of average PSNR differences between rd-curves. *ITU SG16 Doc. VCEG-M33*, 2001.

[6] J. M. Boyce, R. Doré, A. Dziembowski, J. Fleureau, J. Jung, B. Kroon, B. Salahieh, V. K. M. Vadakital, and L. Yu. MPEG immersive video coding standard. *PIEEE*, 109(9):1521–1536, 2021.

[7] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm. Overview of the versatile video coding (VVC) standard and its applications. *IEEE TCSVT*, 31(10):3736–3764, 2021.

[8] A. Cao and J. Johnson. HexPlane: A fast representation for dynamic scenes. In *CVPR*, pages 130–141, 2023.

[9] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. TensorRF: Tensorial radiance fields. In *ECCV*, pages 333–350. Springer, 2022.

[10] Y. Chen, Q. Wu, M. Li, W. Lin, M. Harandi, and J. Cai. Fast feedforward 3d gaussian splatting compression. In *ICLR*, 2025.

[11] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai. HAC: Hash-grid assisted context for 3d gaussian splatting compression. In *ECCV*, 2024.

[12] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai. HAC++: Towards 100x compression of 3d gaussian splatting. *PAMI*, 2025.

[13] Z. Chen, Z. Chen, W. Jiang, W. Wang, L. Liu, and D. Xu. 4DGS-CC: A Contextual Coding Framework for 4D Gaussian Splatting Data Compression, 2025.

[14] W. O. Cho, I. Cho, S. Kim, J. Bae, Y. Uh, and S. J. Kim. 4d scaffold gaussian splatting with dynamic-aware anchor growing for efficient and high-fidelity dynamic scene reconstruction. In *AAAI*, volume 40(5), pages 3363–3371, 2026.

[15] S. Durvasula, S. Muhunthan, Z. Moustafa, R. Chen, R. Liang, Y. Guan, N. Ahuja, N. Jain, S. Panneer, and N. Vijaykumar. ContraGS: Codebook-Condensed and Trainable Gaussian Splatting for Fast, Memory-Efficient Reconstruction. In *ICCV*, pages 28935–28945, 2025.

[16] T. Fan, L. Gao, Y. Xu, Z. Li, and D. Wang. D-DPCC: Deep dynamic point cloud compression via 3d motion prediction. In *IJCAI*, pages 898–904, 2022.

[17] G. Fang, Q. Hu, L. Wang, and Y. Guo. ACRF: Compressing explicit neural radiance fields via attribute compression. In *ICLR*, 2024.

[18] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa. K-Planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, pages 12479–12488, 2023.

[19] G. Gao, A. Azzarelli, H. M. Kwan, N. Anantrasirichai, F. Zhang, W. Andrew, O. Moolan-Feroze, and D. Bull. BVI-CR: A multi-view human dataset for volumetric video compression. In *ISCAS*, pages 1–5. IEEE, 2025.

[20] Q. Gao, J. Meng, C. Wen, J. Chen, and J. Zhang. HiCoM: Hierarchical coherent motion for streamable dynamic scene with 3d gaussian splatting. In *NeurIPS*, 2024.

[21] S. Girish, T. Li, A. Mazumdar, A. Shrivastava, D. Luebke, and S. De Mello. QUEEN: Quantized efficient encoding of dynamic gaussians for streaming free-viewpoint videos. In *NeurIPS*, 2024.

[22] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein. Pup 3DGS: Principled uncertainty pruning for 3d gaussian splatting. In *CVPR*, pages 5949–5958, 2025.

[23] C.-Y. Ho, H.-B. Yang, J.-C. Chiang, Y.-L. Liu, and W.-H. Peng. TED-4DGS: Temporally activated and embedding-based deformation for 4DGS compression. In *WACV*, pages 6371–6380, 2026.

[24] Q. Hu, Z. Zheng, H. Zhong, S. Fu, L. Song, X. Zhang, G. Zhai, and Y. Wang. 4DGC: Rate-aware 4d gaussian compression for efficient streamable free-viewpoint video. In *CVPR*, 2025.

[25] Q. Hu, H. Zhong, Z. Zheng, X. Zhang, Z. Cheng, L. Song, G. Zhai, and Y. Wang. VRVVC: Variable-rate nerf-based volumetric video compression. In *AAAI*, volume 39(4), pages 3563–3571, 2025.

[26] E. S. Jang, M. Preda, K. Mammou, A. M. Tourapis, J. Kim, D. B. Graziosi, S. Rhyu, and M. Budagavi. Video-based point-cloud-compression standard in MPEG: From evidence collection to committee draft [Standards in a nutshell]. *IEEE Signal Processing Magazine*, 36(3):118–123, 2019.

[27] S. Javed, A. J. Khan, C. Dumery, C. Zhao, and M. Salzmann. Temporally compressed 3d gaussian splatting for dynamic scenes. In *BMVC*, 2025.

[28] Z. Jiang, G. Wang, G. K. L. Tam, C. Song, F. W. B. Li, and B. Yang. An end-to-end dynamic point cloud geometry compression in latent space. *Displays*, 80:102528, 2023.

[29] 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 *ICCV*, pages 3334–3342, 2015.

[30] H. Joo, T. Simon, et al. Panoptic studio: A massively multiview system for social interaction capture. *PAMI*, 2017.

[31] G. Kang, Y. Lee, S. Oh, and E. Park. CodecNeRF: Toward fast encoding and decoding, compact, and high-quality novel-view synthesis. In *AAAI*, volume 39(4), pages 4203–4211, 2025.

[32] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d Gaussian splatting for real-time radiance field rendering. *TOG*, 42(4):1–14, 2023.

[33] H. M. Kwan, G. Gao, F. Zhang, A. Gower, and D. Bull. NVRC: Neural video representation compression. *NeurIPS*, 37:132440–132462, 2024.

[34] H. M. Kwan, F. Zhang, A. Gower, and D. R. Bull. Immersive video compression using implicit neural representations. In *PCS*, pages 1–5. IEEE, 2024.

[35] J. C. Lee, J. H. Ko, and E. Park. Optimized minimal 3D Gaussian splatting. In *NeurIPS*, 2025.

[36] M. Lee, B. Lee, L. Y. Lee, E. Lee, J. Y. Jeong, S. Kim, S. Song, J. C. Lee, J. H. Ko, J. Park, and E. Park. Optimized minimal 4D Gaussian splatting for efficient dynamic scene representation. In *NeurIPS*, 2026.

[37] S. Lee, F. Shu, Y. Sanchez, T. Schierl, and C. Hellge. Ecrf: Entropy-constrained neural radiance fields compression with frequency domain optimization. In *MMSP*, pages 1–6. IEEE, 2024.

[38] S. Lee, F. Shu, Y. Sanchez, T. Schierl, and C. Hellge. Compression of 3d gaussian splatting with optimized feature planes and standard video codecs. In *ICCV*, pages 25496–25505, 2025.

[39] G. Li, W. Gao, and W. Gao. MPEG geometry-based point cloud compression (G-PCC) standard. In *Point Cloud Compression: Technologies and Standardization*, pages 135–165. Springer, 2024.

[40] G. Li, W. Gao, and W. Gao. MPEG video-based point cloud compression (V-PCC) standard. In *Point Cloud Compression: Technologies and Standardization*, pages 199–218. Springer, 2024.

[41] H. Li, S. Li, X. Gao, A. Batuer, L. Yu, and Y. Liao. GIFStream: 4d gaussian-based immersive video with feature stream. In *CVPR*, pages 21761–21770, 2025.

[42] L. Li, Z. Shen, Z. Wang, L. Shen, and L. Bo. Compressing volumetric radiance fields to 1 mb. In *CVPR*, pages 4222–4231, 2023.

[43] S. Li, H. Li, Y. Liao, and L. Yu. Nerfcodec: Neural feature compression meets neural radiance fields for memory-efficient scene representation. In *CVPR*, pages 21274–21283, 2024.

[44] S. Li, C. Wu, H. Li, X. Gao, Y. Liao, and L. Yu. Gscodec studio: A modular framework for gaussian splat compression. *IEEE TCSVT*, 2026.

[45] T. Li, M. Slavcheva, M. Zollhoefer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, R. Newcombe, et al. Neural 3d video synthesis from multi-view video. In *CVPR*, pages 5521–5531, 2022.

[46] Z. Li, Z. Chen, Z. Li, and Y. Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, pages 8508–8520, 2024.

[47] H. Lin, S. Chen, J. H. Liew, D. Y. Chen, Z. Li, Y. Zhao, S. Peng, H. Guo, X. Zhou, G. Shi, J. Feng, and B. Kang. Depth anything 3: Recovering the visual space from any views. In *ICLR*, 2026.

[48] Q. Ling, Z. Cheng, D. Feng, S. Wang, C. Zhu, G. Lu, H. Sun, J. Katto, and L. Song. A multi-grid implicit neural representation for multi-view videos. *arXiv preprint arXiv:2509.16706*, 2025.

[49] L. Liu, Z. Chen, W. Jiang, W. Wang, and D. Xu. HEMGS: A hybrid entropy model for 3d gaussian splatting data compression. *arXiv preprint arXiv:2411.18473*, 2024.

[50] M. Liu, Q. Yang, H. Huang, W. Huang, Z. Yuan, Z. Li, Y. Xu, and Y. Guan. Light4GS: Lightweight compact 4d gaussian splatting generation via context model. *IEEE TCSVT*, 2026.

[51] X. Liu, B. Chen, Z. Liu, Y. Wang, and S.-T. Xia. An exploration with entropy constrained 3D Gaussians for 2D video compression. In *ICLR*, 2025.

[52] X. Liu, X. Wu, S. Wang, Z. Li, and S. Kwong. CompGS++: Compressed gaussian splatting for static and dynamic scene representation. *arXiv preprint arXiv:2504.13022*, 2025.

[53] Z. Liu, R. Song, Y. Huang, Y. Hu, X. Zhang, J. Shao, Z. Lin, and J. Zhang. Feed-Forward 3D Gaussian Splatting Compression with Long-Context Modeling. *arXiv preprint arXiv:2512.00877*, 2025.

[54] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai. Scaffold-GS: Structured 3d gaussians for view-adaptive rendering. In *CVPR*, pages 20654–20664, 2024.

[55] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, pages 800–809. IEEE, 2024.

[56] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[57] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG*, 41(4):1–15, 2022.

[58] K. Navaneet, K. Pourahmadi Meibodi, S. Abbasi Koohpayegani, and H. Pirsiavash. CompGS: Smaller and faster gaussian splatting with vector quantization. In *ECCV*, pages 330–349. Springer, 2024.

[59] M. Quach, G. Valenzise, and F. Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *ICIP*, pages 4320–4324, 2019.

[60] D. Rho, B. Lee, S. Nam, J. C. Lee, J. H. Ko, and E. Park. Masked wavelet representation for compact neural radiance fields. In *CVPR*, pages 20680–20690, 2023.

[61] S. Rota Bulò, L. Porzi, and P. Kontschieder. Revising densification in gaussian splatting. In *ECCV*, pages 347–362. Springer, 2024.

[62] N. Sabater, G. Boisson, B. Vandame, P. Kerbiriou, F. Babon, M. Hog, R. Gendrot, T. Langlois, O. Bureller, A. Schubert, V. Allie, and P. Pérez. Dataset and pipeline for multi-view light-field video. In *CVPRW*, pages 30–40, 2017.

[63] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuca, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko. Emerging MPEG standards for point cloud compression. *JETCAS*, 9(1):133–148, 2019.

[64] X. Sheng, Y. Zhang, L. Xu, and S. Wang. DCVC-MV: Deep contextual multiview video compression with efficient inter-view prediction. *IEEE TMM*, 2026.

[65] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 33:7462–7473, 2020.

[66] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE TCSVT*, 22(12):1649–1668, 2012.

[67] J. Sun, H. Jiao, G. Li, Z. Zhang, L. Zhao, and W. Xing. 3DGStream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *CVPR*, 2024.

[68] J. Tang, X. Chen, J. Wang, and G. Zeng. Compressible-composable NeRF via rank-residual decomposition. *NeurIPS*, 35:14798–14809, 2022.

[69] S. Teng, G. Gao, D. Danier, Y. Jiang, F. Zhang, N. Anantrasirichai, T. Davis, Z. Liu, and D. Bull. Gfix: Perceptually enhanced gaussian splatting video compression. In *ISCAS*, pages 4829–4833. IEEE, 2026.

[70] H. Wang, M. H. Vali, and A. Solin. Compressing 3d gaussian splatting by noise-substituted vector quantization. In *Scandinavian Conference on Image Analysis*, pages 338–352. Springer, 2025.

[71] L. Wang, Q. Hu, Q. He, Z. Wang, J. Yu, T. Tuytelaars, L. Xu, and M. Wu. Neural residual radiance fields for streamably free-viewpoint videos. In *CVPR*, pages 76–87, 2023.

[72] Y. Wang, Z. Li, L. Guo, W. Yang, A. C. Kot, and B. Wen. ContextGS: Compact 3d gaussian splatting with anchor level context model. *NeurIPS*, 37:51532–51551, 2024.

[73] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *CVPR*, pages 20310–20320, June 2024.

[74] M. Wu, Z. Wang, G. Kouros, and T. Tuytelaars. TeTriRF: Temporal tri-plane radiance fields for efficient free-viewpoint video. In *CVPR*, pages 6487–6496, 2024.

[75] H.-B. Yang, M.-Z. He, C.-W. Yang, J.-C. Chiang, Y.-L. Liu, and W.-H. Peng. IC-4DGS: Illumination-compensated 4d gaussian splatting under photometric variations. In *ICIP*, pages 1–6. IEEE, 2026.

[76] Y.-T. Zhan, C.-Y. Ho, H. Yang, Y.-H. Chen, J. C. Chiang, Y.-L. Liu, and W.-H. Peng. CAT-3DGS: A context-adaptive triplane approach to rate-distortion-optimized 3DGS compression. In *ICLR*, 2025.

[77] 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.

[78] W. Zhang, Y. Zhao, Q. Wang, Z. Xu, L. Song, and Z. Cheng. D-FCGS: Feedforward compression of dynamic gaussian splatting for free-viewpoint videos. In *AAAI*, volume 40(19), pages 16361–16369, 2026.

[79] X. Zhang, Z. Liu, Y. Zhang, X. Ge, D. He, T. Xu, Y. Wang, Z. Lin, S. Yan, and J. Zhang. MEGA: Memory-efficient 4d gaussian splatting for dynamic scenes. In *ICCV*, pages 27828–27838, 2025.

[80] Z. Zhang, G. Lu, H. Liang, Z. Cheng, A. Tang, and L. Song. Rate-aware compression for neRF-based volumetric video. In *ACM MM*, 2024.

[81] T. Zhao, J. Chen, C. Leng, and J. Cheng. TinyNeRF: Towards 100 x compression of voxel radiance fields. In *AAAI*, volume 37(3), pages 3588–3596, 2023.

[82] Z. Zheng, Z. Wu, H. Zhong, Y. Tian, N. Cao, L. Xu, J. Yao, X. Zhang, Q. Hu, and W. Zhang. 4DGCPro: Efficient hierarchical 4D Gaussian compression for progressive volumetric video streaming. In *NeurIPS*, 2025.

[83] Z. Zheng, H. Zhong, Q. Hu, X. Zhang, L. Song, Y. Zhang, and Y. Wang. HPC: Hierarchical progressive coding framework for volumetric video. In *ACM MM*, pages 7937–7946, 2024.

[84] H. Zhong, Z. Zheng, Q. Hu, Y. Tian, N. Cao, L. Xu, X. Zhang, Z. Cheng, L. Song, and W. Zhang. 4D-MoDe: Towards editable and scalable volumetric streaming via motion-decoupled 4d gaussian compression. *arXiv preprint arXiv:2509.17506*, 2025.

[85] C. Zhu, G. Lu, B. He, R. Xie, and L. Song. Implicit-explicit integrated representations for multi-view video compression. *TIP*, 34:1106–1118, 2025.

A Broader Impact

This work improves the efficiency of volumetric video compression, with potential benefits for telepresence, remote collaboration, education, cultural preservation, digital media production, and immersive communication. By reducing storage and transmission costs while retaining real-time decoding, SAGA may make high-quality immersive media more accessible under limited bandwidth and compute budgets, and may reduce the infrastructure and energy costs of large-scale volumetric video delivery.

At the same time, more compact and realistic volumetric representations may increase risks related to privacy, unauthorized capture, surveillance, and deceptive or non-consensual synthetic media, especially when the content contains identifiable people or private environments. Responsible deployment should therefore consider consent, access control, provenance tracking, watermarking or authentication, and dataset governance. Our work focuses on compression and reconstruction quality on established research datasets, and does not develop tools for identity manipulation, text-driven content synthesis, or surveillance.

B Dataset/Code License

We list the licenses of all datasets and the code used in this paper in Table B.1.

Table B.1: **Existing assets used in this work.** We list the datasets, codebases, codecs, and models used in our experiments, together with their sources and license or usage terms where available.

Asset	Type	Source / Citation	License / Terms
Neu3D	Dataset	[45]	CC BY-NC 4.0
Panoptic Sports / Panoptic Studio	Dataset	[30]	Research-only, non-commercial
MPEG MIV CTC sequences	Dataset	[6]	Per-sequence licenses (MIV content database)
TMIV-24.0	Codec / software	[6]	BSD 3-Clause
HM-18.0	Codec / software	[66]	BSD 3-Clause
VVenC	Codec / software	[7]	BSD 3-Clause Clear (no patent grant)
Depth Anything 3 (DA3-Large)	Code / model weights	[47]	DA3-Large weights: CC BY-NC 4.0
3DGStream	Code / method	[67]	MIT
4D Scaffold GS	Code / method	[14]	Gaussian-Splatting License (non-commercial)
4DGC	Code / method	[24]	No explicit license
STG	Code / method	[46]	MIT; bundled 3DGS under Gaussian-Splatting License
GIFStream	Code / method	[41]	Scaffold-GS parts under Gaussian-Splatting License
D-FCGS	Code / method	[78]	Not specified
OMG4	Code / model weights	[36]	No explicit license
GSCodec Studio	Code / method	[44]	Apache-2.0
TED-4DGS	Code / method	[23]	No explicit license
IC-4DGS	RD results	[75]	RD results provided by the authors
LPIPS	Evaluation metric / code	[77]	BSD 2-Clause

C Detailed Configurations

C.1 SAGA Configurations

In Table C.2 we provide the detailed model configurations for the proposed SAGA.

C.2 Dataset Preparation

MPEG MIV. Following the MPEG Immersive Video Common Test Conditions (CTC) [6], we select five sequences from the MIV main-anchor configuration: *ClassroomVideo (A01)*, *Chess (B02)*, *Painter (D01)*, *Frog (E01)*, and *Carpark (E02)*. These sequences comprise two computer-generated and three natural scenes, covering diverse scene content and camera configurations, including omnidirectional (equirectangular) and perspective source views with 9 to 16 cameras. We use the first 65 frames of each sequence. This selection follows the standardized CTC benchmark to evaluate our method on representative multi-view video content.

C.3 Baseline Configurations

MIV Test Model. The MPEG Immersive Video Test Model (TMIV) [6] is the reference software for MPEG Immersive Video. It compresses multi-view immersive content by exploiting inter-view

Table C.2: **Model and training configurations.** Model-size-dependent entries are listed in S / M / L order; all other settings are shared.

Module	Configuration	Value (S / M /L)
Hierarchical 4D anchor scaffold		
	Coarse-anchor budget	6k / 12k / 24k
	Fine-anchor budget, N_{budget}	45k / 90k / 180k
	Fine anchors per coarse anchor, $N_{\min}/N_{\max}$	1 / 32
	Maximum tree depth	14
	Temporal and statistic scaling, $\lambda_\tau, \lambda_{\boldsymbol{h}}$	6.0 / 0.1
	Anchor-allocation exponents, α, β	0.5 / 0.5
	Local-offset boundary, λ_A	1.0 / 0.01
Kernel interpolation		
	4D kNN candidates and neighbors, M, K	12 / 8
	Support radii, $\rho_{\mathrm{c}}, \rho_{\mathrm{f}}$	3.0 / 3.0
	Kernel temperatures, $T_{\mathrm{c}}, T_{\mathrm{f}}$	1.0 / 1.0
	Temporal distance weight, λ_t	1.0
	Interpolation MLP width, $\mathrm{MLP}_{\mathrm{c}}, \mathrm{MLP}_{\mathrm{f}}$ (depth 3)	64 / 96 / 128
Primitive mask		
	Mask threshold on $\mathrm{sigmoid}$ logit	0.01
	Mask loss weight, λ_{mask}	1.0×10^{-3}
Hierarchical entropy model		
	Coarse / fine-anchor / primitive coding groups	16 / 1 / 16
	INR hyperprior depth / width	3 / 16
	Memory slots and slot dimension, L	64 / 16
	Memory read (single-head attention) dimension	16
	Entropy-parameter network depth / width	3 / 16
	Batch-token embedding depth / width, aggregation	3 / 16, mean
Optimization and coding		
	Proposal warm-up (per 50-frame GOP) / compression iterations	4k / 30k
	Quantization start / rate warm-up / quant-noise stage	10k / 2k / last 3k
	Initial quantization precision	8 bits
	Rate weight λ_r (per MB)	0.0003, 0.004, 0.015, 0.06, 0.001
	Optimizer and initial learning rate	Adam / 10^{-3}–10^{-2}
	Learning-rate schedule	Exp. decay to 1%
	Rendering SSIM weight, η	0.2

redundancy and packing pruned texture and geometry information into video atlases. We use TMIV 24.0 with the MIV Main anchor configuration, in which depth maps are required as geometry inputs and are compressed together with texture; we therefore report the size of the complete multiplexed MIV bitstream, which contains the texture and geometry atlases as well as the atlas metadata. For the MPEG MIV sequences, we use the first 65 frames of A01, B02, D01, E01 and E02 with the supplied ground-truth depth maps and the anchor QPs of the common test conditions. For Neu3D and Panoptic Sports, RGB frames are converted to 10-bit YUV 4:2:0 (BT.709, limited range) and texture and geometry are coded with QPs $\{28, 34, 40, 46\}$ and $\{6, 10, 16, 22\}$, respectively, for RP1–RP4. We invoke TMIV through its `encode.py` script, which runs the TMIV encoder, the selected video encoder for each atlas component, the TMIV multiplexer, and the parser/bitrate report.

```
python3 <tmiv_root>/scripts/encode.py \
  -i <input_yuv_root> \
  -o <output_root> \
  -s <content_id> \
  -n <frame_count> \
  -f 0 \
  -c <tmiv_root>/config/ctc/miv_main_anchor/A_1_TMIV_encode.json \
  -m <tmiv_root>/config/ctc/miv_main_anchor/A_3_TMIV_mux.json \
  -q <qp_file> \
  -v VVenC \
```

```
  -C <tmiv_install_dir>/share/config/vvenc/randomaccess_medium.cfg \
  -r <rate_point> \
  -t <tmiv_install_dir> \
  --config-dir <tmiv_config_dir> \
  -j <threads>
```

The synthesized test views used for distortion evaluation are decoded from the TMIV bitstream, while the best-reference views are rendered directly from the source views for metric computation:

```
<tmiv_install_dir>/bin/TmivDecoder \
  -c <tmiv_root>/config/ctc/miv_main_anchor/A_4_TMIV_decode.json \
  -p configDirectory <tmiv_config_dir> \
  -p inputDirectory <output_root> \
  -p outputDirectory <output_root> \
  -s <content_id> -n <frame_count> -N <output_frame_count> \
  -r <rate_point> -P <pose_trace_id> -j <threads>

<tmiv_install_dir>/bin/TmivRenderer \
  -c <tmiv_root>/config/ctc/best_reference/R_1_TMIV_render.json \
  -p configDirectory <tmiv_config_dir> \
  -p inputDirectory <input_yuv_root> \
  -p outputDirectory <output_root> \
  -s <content_id> -n <frame_count> -N <output_frame_count> \
  -f 0 -r RP0 -P <pose_trace_id> -j <threads>
```

Internally, when VVenC is selected, each atlas video component is encoded by a command of the following form:

```
vvencFFapp \
  -c <video_encoder_config_file> \
  -i <atlas_yuv_file> \
  -b <video_bitstream_file> \
  -s <width>x<height> \
  -q <qp> \
  -f <frame_count> \
  -fr <frame_rate> \
  --InputBitDepth 10 \
  --OutputBitDepth 10 \
  --CbQpOffset <chroma_qp_offset> \
  --CrQpOffset <chroma_qp_offset> \
  --Threads <threads>
```

Depth Estimation. Neu3D and Panoptic Sports do not provide ground-truth depth. We estimate per-view depth maps using Depth Anything 3 (DA3-Large) [47], running pose-conditioned multi-view inference over all source views of each timestamp so that the predicted depths share the scale of the given camera extrinsics:

```
pred = model.inference(
    image=[view["images"][t] for view in source_views],
    intrinsics=intrinsics,
    extrinsics=extrinsics,
    align_to_input_ext_scale=True,
    process_res=504,
    process_res_method="upper_bound_resize",
)
```

The held-out test views (Neu3D `cam00` and the Panoptic test cameras) are excluded from inference. The predicted depths are resampled to the texture resolution with nearest-neighbour sampling and quantized to 16-bit inverse depth, with a fixed near/far range per sequence ($0.9\times$ the minimum and $1.1\times$ the maximum predicted depth); the value 0 is reserved for invalid samples:

```
inv = (
    (1 / np.clip(z, near, far) - 1 / far)
```

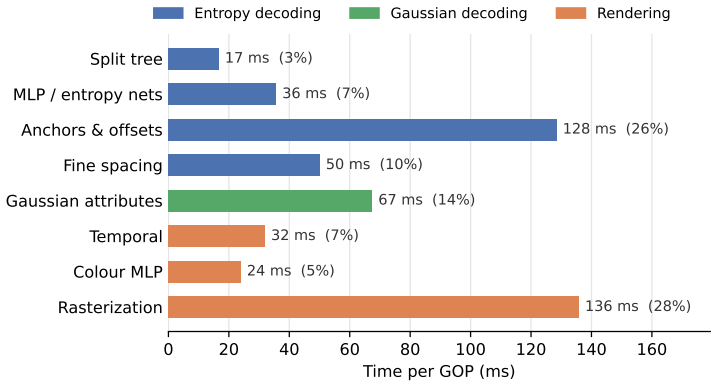


Figure D.1: **Inference speed breakdown**.

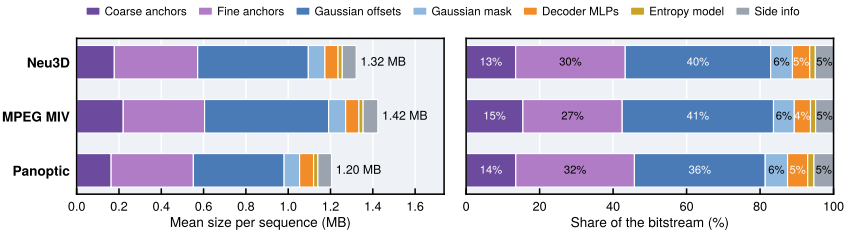


Figure D.2: **Bitstream breakdown**.

```
    / (1 / near - 1 / far)
)
q = np.clip(np.rint(inv * 65535), 1, 65535).astype(np.uint16)
```

D Additional Experimental Results

D.1 Bitstream and Inference Speed Breakdown

Inference speed breakdown. As shown in Fig. D.1, rasterization (28%) and anchor/offset decoding (26%) dominate the inference time, accounting for more than half of the total runtime. In contrast, entropy decoding and other lightweight operations introduce relatively limited overhead.

Bitstream breakdown. As shown in Fig. D.2, Gaussian-related parameters account for the majority of the bitstream across different datasets, with Gaussian offsets contributing the largest share (approximately 41–46%). This suggests that Gaussian parameter representation, particularly offsets, remains the primary bottleneck for further compression.

D.2 Raw Values

In Table D.3 we provide the per-sequence raw values for the RD-curves of SAGA on the Neu3D and MPEG MIV benchmark.

D.3 Results on Challenging Sequences

Table D.4 reports the uncompressed representations (the setting of Table 1) on the most challenging sequences of each dataset: *coffee_martini* and *flame_salmon* from Neu3D, which contain specular surfaces, fast hand motion and thin flames; Frog (E01) from MPEG MIV, with large non-rigid motion close to the cameras; and both Panoptic Sports sequences, with fast full-body motion. On Panoptic Sports and E01, SAGA outperforms 4D Scaffold-GS by 1.6–2.4 dB in PSNR, and on Panoptic Sports even the small configuration (8.71 MB) is 1.7–1.9 dB better at 13–31× smaller size. On

Table D.3: **Rate-distortion results on Neu3D and MPEG MIV.**

(a) Neu3D

Sequence	QP	Size (MB)	PSNR	SSIM	LPIPS	Sequence	QP	Size (MB)	PSNR	SSIM	LPIPS
coffee_martini	QP_1	2.625	27.476	0.9092	0.1331	flame_salmon	QP_1	2.516	27.655	0.9074	0.1135
	QP_2	2.063	27.331	0.8931	0.1391		QP_2	2.933	28.207	0.9165	0.0917
	QP_3	1.775	27.451	0.8983	0.1240		QP_3	1.843	28.084	0.9114	0.1030
	QP_4	1.379	27.026	0.8851	0.1564		QP_4	1.433	28.056	0.9094	0.1048
	QP_5	1.020	26.390	0.8717	0.1859		QP_5	1.092	27.137	0.8875	0.1457
cook_spinach	QP_1	4.259	33.956	0.9565	0.0450	flame_steak	QP_1	2.613	33.864	0.9574	0.0477
	QP_2	3.255	33.742	0.9558	0.0460		QP_2	3.035	34.462	0.9603	0.0386
	QP_3	1.837	33.110	0.9504	0.0583		QP_3	1.909	34.496	0.9587	0.0430
	QP_4	1.336	32.703	0.9467	0.0642		QP_4	1.241	33.748	0.9532	0.0564
	QP_5	1.015	31.414	0.9316	0.0960		QP_5	0.988	32.159	0.9397	0.0859
cut_roasted_beef	QP_1	3.974	34.538	0.9579	0.0457	sear_steak	QP_1	3.197	34.898	0.9632	0.0377
	QP_2	3.073	34.454	0.9571	0.0481		QP_2	3.043	34.997	0.9625	0.0367
	QP_3	2.088	34.115	0.9543	0.0530		QP_3	1.933	34.637	0.9603	0.0412
	QP_4	1.344	33.343	0.9472	0.0676		QP_4	1.250	33.936	0.9547	0.0520
	QP_5	1.013	31.808	0.9325	0.0969		QP_5	0.974	32.475	0.9415	0.0817

(b) MPEG MIV

Sequence	QP	Size (MB)	PSNR	SSIM	LPIPS	Sequence	QP	Size (MB)	PSNR	SSIM	LPIPS
Painter (D01)	QP_1	3.151	36.093	0.9621	0.0307	Frog (E01)	QP_1	4.481	28.254	0.9006	0.1067
	QP_2	2.713	36.119	0.9621	0.0305		QP_2	3.746	28.217	0.8991	0.1102
	QP_3	1.896	35.477	0.9578	0.0340		QP_3	2.769	27.843	0.8927	0.1153
	QP_4	1.287	34.272	0.9458	0.0445		QP_4	1.748	26.700	0.8711	0.1386
	QP_5	0.959	32.281	0.9178	0.0719		QP_5	1.243	24.090	0.7953	0.2177
Carpark (E02)	QP_1	3.185	33.571	0.9332	0.0455						
	QP_2	2.601	33.445	0.9310	0.0475						
	QP_3	1.915	32.995	0.9252	0.0531						
	QP_4	1.326	31.814	0.9053	0.0732						
	QP_5	1.013	29.800	0.8583	0.1211						

the two Neu3D sequences, the large configuration is within 0.04 dB (*coffee_martini*) and 0.34 dB (*flame_salmon*) of 4D Scaffold-GS in PSNR while being 5–8× smaller.

Table D.4: **Challenging sequences.** Uncompressed representations; sizes are raw (fp32) model sizes. The size of SAGA is fixed by its anchor budgets and is the same for every sequence. Best PSNR and SSIM per sequence in bold.

		4D Scaffold-GS [14]			SAGA-S (8.71 MB)		SAGA-M (17.37 MB)		SAGA-L (34.54 MB)	
Dataset	Sequence	Size (MB)↓	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Neu3D	*coffee_martini*	184.8	**28.65**	**0.919**	27.27	0.893	28.34	0.912	28.61	0.918
	flame_salmon	263.3	**28.95**	**0.926**	28.03	0.911	28.37	0.915	28.61	0.922
MPEG MIV	Frog (E01)	310.8	28.72	0.923	27.43	0.883	28.46	0.903	**30.33**	**0.935**
Panoptic Sports	*basketball*	269.9	27.99	0.922	29.72	0.930	**29.94**	**0.932**	29.52	0.931
	boxes	117.4	27.89	0.928	29.75	0.937	**30.31**	**0.940**	28.82	0.935

E Additional Qualitative Results

Here we provide exemplary apple-to-apple comparisons between the source frames and reconstructions. More visual comparisons with different camera motions are available on the **project page**.



Figure E.3: Qualitative comparison on Neu3D *sear_steak*, frame 150.



Figure E.4: Qualitative comparison on Neu3D *cut_roasted_beef*, frame 150.



Figure E.5: Qualitative comparison on MPEG MIV *E01*, frame 32.

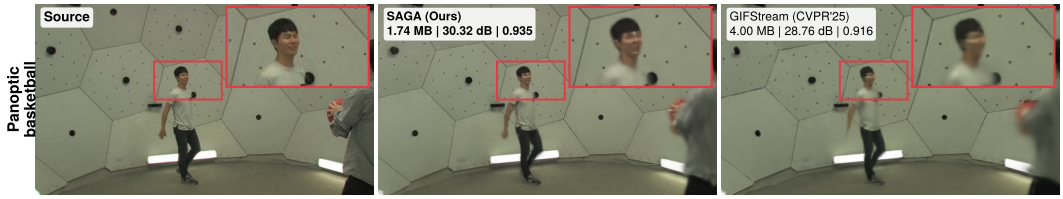


Figure E.6: Qualitative comparison on Panoptic *basketball*, frame 75.

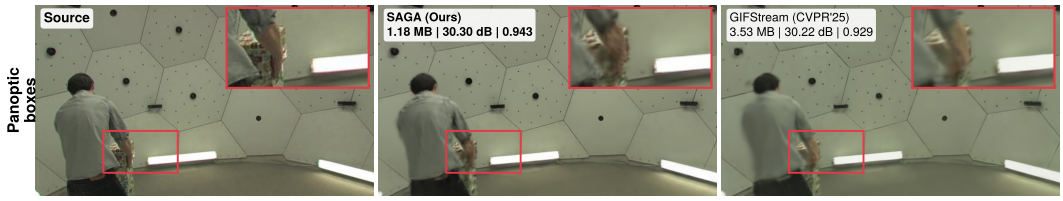


Figure E.7: Qualitative comparison on Panoptic *boxes*, frame 75.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The introduction states the main contributions of SAGA, including hierarchical sparse 4D anchors, inter-anchor interpolation, residual refinement, and long-context entropy modeling, and the experimental claims are tied to Neu3D, Panoptic Sports, and MPEG MIV benchmarks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a limitations discussion noting per-scene optimization cost, possible difficulty with fast motion, topology changes, sparse camera coverage, and future directions such as faster encoding and adaptive anchor allocation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results, theorems, or proofs; the equations are methodological definitions of the representation, entropy model, rendering process, and training objective.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the datasets, evaluation protocols, baselines, metrics, and major implementation components needed to reproduce the main results; additional hyperparameters and training details can be provided in the appendix or supplemental material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The experiments use publicly available datasets and standard baselines, and the authors will provide code, configuration files, and instructions for reproducing the main results in the supplemental material or public release.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the datasets, train/test camera splits where applicable, input resolutions, baselines, evaluation metrics, decoding platform, and comparison settings used for novel view synthesis and volumetric video compression.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports averaged quantitative results and BD-rate comparisons, but does not currently include error bars, confidence intervals, or statistical significance tests over multiple random seeds or repeated runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports training time and decoding speed, and states that runtime is measured on a single NVIDIA RTX 3090 GPU; full per-sequence compute and memory details can be included in the appendix or supplemental material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work focuses on compression of established research datasets and does not involve deceptive data collection, human-subject studies, or unsafe model release; the authors have considered privacy and misuse risks in the broader impact discussion.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts such as efficient immersive communication and reduced storage/transmission cost, as well as potential negative impacts related to privacy, surveillance, unauthorized volumetric capture, and misuse of realistic 3D media.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk generative model, scraped dataset, or identity-manipulation system; the method is a volumetric video compression framework evaluated on established datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the datasets, codecs, and prior methods used in the experiments; dataset access terms, code licenses, and baseline versions should be explicitly listed in the appendix or supplemental material.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper does not introduce a new dataset or benchmark asset; if code or trained models are released, they should be documented separately with usage instructions and license information.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or newly collected human-subject data.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human-subject experiments or new data collection from participants, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method does not use LLMs as an algorithmic component; any LLM use limited to writing, editing, or formatting does not require declaration under the stated policy.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.