Optimized Minimal 4D Gaussian Splatting for Efficient Dynamic Scene Representation

Minseo Lee[1*] **Byeonghyeon Lee**[1*] **Lucas Yunkyu Lee**[2,3] **Eunsoo Lee**[4]
JunYoung Jeong[5] **Sangmin Kim**[2] **Seunghyeon Song**[4] **Joo Chan Lee**[4]
Jong Hwan Ko[4] **Jaesik Park**[2] **Eunbyung Park**[1†]

[1] Yonsei University [2] Seoul National University [3] POSTECH [4] Sungkyunkwan University
[5] Electronics and Telecommunications Research Institute
https://minshirley.github.io/OMG4/

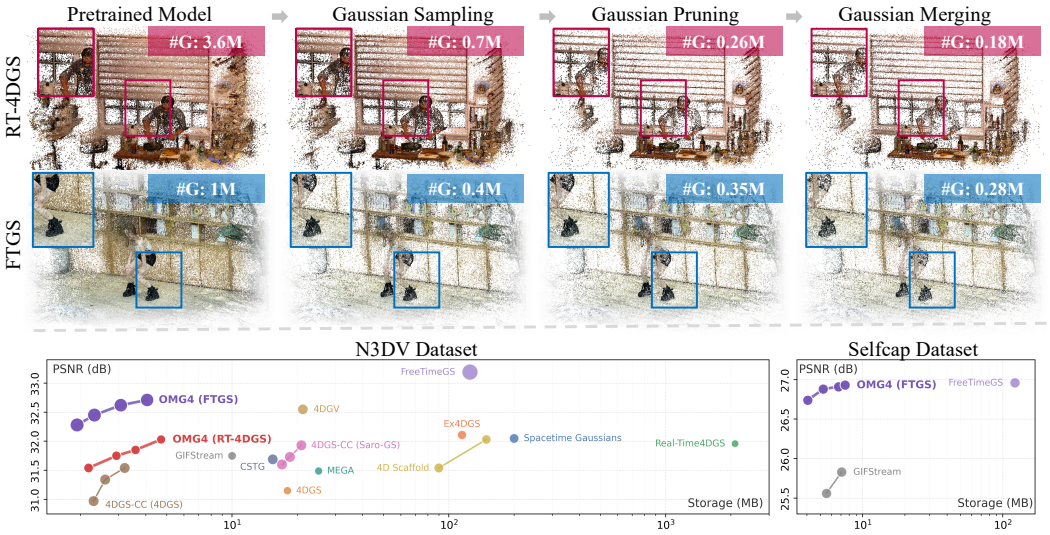


Figure 1: The overall *OMG4* pipeline and performance comparison. Top: Multi-stage compression applied to Real-Time4DGS (RT-4DGS) on *Cook Spinach* (N3DV dataset) and FreeTimeGS (FTGS) on *Dance1* (Selfcap dataset). Bottom: Rate-distortion comparison across 4DGS methods (larger circles indicate higher FPS).

Abstract

4D Gaussian Splatting has emerged as a new paradigm for dynamic scene representation, enabling real-time rendering of scenes with complex motions. However, it faces a major challenge of storage overhead, as millions of Gaussians are required for high-fidelity reconstruction. Compressing explicit 4D Gaussians is challenging because they exhibit heterogeneous redundancy across static and dynamic regions, and their temporal attributes make direct quantization unstable. In this work, we present *OMG4* (Optimized Minimal 4D Gaussian Splatting), a framework that constructs a compact set of salient Gaussians capable of faithfully representing 4D Gaussian models. Our method progressively reduces Gaussians in three stages: (1) Gaussian Sampling to identify primitives critical to reconstruction fidelity, (2) Gaussian Pruning to remove redundancies, and (3) Gaussian Merging to fuse primitives

*Equal contribution
†Corresponding author

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

with similar characteristics. In addition, we integrate implicit appearance compression and extend Sub-Vector Quantization (SVQ) to 4D representations with a staged quantization scheme, further reducing storage while preserving quality. Extensive experiments on standard benchmark datasets demonstrate that *OMG4* achieves a favorable rate-distortion trade-off over recent state-of-the-art methods, reducing model sizes by over 60% while maintaining reconstruction quality. These results demonstrate the effectiveness of *OMG4* as a practical framework for compact and high-fidelity 4D scene representation.

1 Introduction

3D Gaussian Splatting [15] has recently achieved remarkable success, becoming a backbone for diverse 3D vision tasks ranging from 3D novel view synthesis to downstream applications such as visual odometry [14, 44], 3D scene editing [19, 40], and degradation-aware rendering [18, 37, 47]. Building on this success, 4D Gaussian representations that explicitly model space and time have emerged as a new paradigm for dynamic scene reconstruction [39, 41, 45, 46]. By augmenting Gaussian primitives with temporal parameters, these approaches can capture object motion and appearance variations over time, enabling photorealistic real-time novel view synthesis for applications such as free-viewpoint video [10, 23, 36], autonomous driving simulation [16, 51], and VR/AR [13, 32, 42].

Existing 4DGS methods typically follow either deformation-based modeling from a canonical 3D Gaussian set [41, 46] or direct optimization of 4D Gaussians in a unified space-time volume [45]. The latter often achieves high fidelity for complex object motions, but remains storage-heavy because millions of primitives with high-dimensional and time-varying attributes are required (e.g., Real-Time4DGS [45] produces millions of 4D Gaussians, consuming over a gigabyte of memory). This overhead limits practicality in real-time, mobile, and streaming scenarios, highlighting the need for effective compression techniques that preserve both visual fidelity and rendering speed.

Compressing explicit 4D Gaussian representations, however, is not a straightforward extension of static 3DGS compression. 4D Gaussian exhibits heterogeneous redundancy across static and dynamic regions, and introduces temporal attributes that are tightly coupled with spatial geometry and appearance, hence directly quantizing all attributes at once leads to unstable optimization. These observations suggest that effective 4DGS compression should jointly address primitive-level redundancy and stable spatio-temporal attribute compression.

Several works have attempted to alleviate the significant storage requirement of explicit 4D representations [23, 48, 49]. 4DGS-1K [48] presents a lifespan-based importance score to prune short-lived Gaussians, reducing the number of primitives and compressing the representation to hundreds of megabytes. GIFStream [23] performs motion-aware pruning using feature streams, while Light4GS [28] and ADC-GS [11] adopt context modeling or anchor-based structures to compress deformation-based dynamic Gaussians [41]. Despite these advances, existing methods still require tens of megabytes to represent only a few seconds of dynamic content, limiting their practicality.

In this paper, we propose *OMG4* (Optimized Minimal 4D Gaussian Splatting), a framework for compact and high-fidelity dynamic scene representation. Starting from a pretrained Real-Time4DGS [45], *OMG4* progressively constructs a compact primitive set through Gaussian Sampling, Gaussian Pruning, and Gaussian Merging, which identify static-dynamic salient primitives, remove redundant ones, and fuse similar primitives in local space-time neighborhoods. Given the resulting compact set, we further compress Gaussian attributes by integrating implicit appearance modeling and extending Sub-Vector Quantization (SVQ) [22] to 4D representations. Importantly, we show that attribute compression becomes tractable only after Gaussian reduction, and introduce a staged 3D-to-4D SVQ scheme that mitigates the instability of simultaneously quantizing spatial and temporal attributes.

We conduct experiments on diverse benchmarks, including N3DV [25], ENeRF-Outdoor [27], and SelfCap [43]. To the best of our knowledge, we achieve state-of-the-art performance, significantly reducing the volume of the baseline model by three orders of magnitude. Notably, compared to GIFStream [23], a recent state-of-the-art approach, our method reduces storage size by approximately 64% (from 10.0 MB to 3.59 MB in the N3DV dataset). We further demonstrate the generality of *OMG4* by applying it to other dynamic scene representations, including STG [26] and FreeTimeGS [39].

To sum up, our contributions are as follows:

- We propose a novel static-dynamic Gaussian minimization framework, progressively reducing the number of Gaussians through Gaussian Sampling, Gaussian Pruning, and Gaussian Merging, while maintaining the reconstruction quality.
- We extend Sub-Vector Quantization (SVQ) to dynamic 4D representations with a staged quantization scheme, and empirically show that such attribute compression becomes tractable only after sufficient Gaussian reduction via our pipeline.
- We validate the generality of our method by applying it to various dynamic scene representation baselines, including Real-Time4DGS [45], STG [26], and FreeTimeGS [39].
- To the best of our knowledge, we achieve a state-of-the-art rate-distortion trade-off, reducing the baseline model from gigabytes to a few megabytes.

2 Related Work

2.1 3D Gaussian Splatting and Compression

3D Gaussian Splatting (3DGS) [15] enables real-time and high-fidelity novel view synthesis with explicit Gaussian primitives, but its large number of primitives leads to substantial storage overhead. Recent 3DGS compression methods reduce this cost through pruning-based strategies [8, 50], learnable masks and vector quantization [21], sub-vector or locality-aware attribute encoding [22, 35], and structured anchor-based representations [5, 30]. Specifically, OMG [22] introduces sub-vector quantization (SVQ) that splits the vectors into multiple small sub-vectors and applies VQ, achieving significant performance improvement. Although these approaches effectively reduce the storage requirements of 3DGS, they do not straightforwardly generalize to 4D representations.

2.2 Dynamic 3D Gaussian Splatting and Compression

Early efforts to extend static scene reconstruction to dynamic settings [4, 9, 24, 34, 38] were primarily based on neural volumetric rendering, but suffered from high computational costs. More recently, many studies have sought to extend 3DGS to dynamic scenes, which can be broadly categorized into two approaches: (1) representing dynamic scenes with 4D Gaussian primitives that jointly encode spatial and temporal dimensions [26, 45], or (2) deforming 3D Gaussians at each timestamp via a deformation field [2, 17, 41, 46]. Real-Time4DGS [45] achieves high-fidelity dynamic reconstruction by parameterizing the 4D volume with a set of 4D Gaussians. Recently, FreeTimeGS [39] has shown promising performance by moving 3D Gaussians over time, leveraging motion vectors.

Similar to 3DGS, dynamic extensions also require a significant number of primitives, motivating the development of lightweight methods. Within the first category, which employs 4D Gaussian primitives, CSTG [20], MEGA [49], and 4DGS-1K [48] have been proposed. Among them, 4DGS-1K [48] reduces storage by discarding Gaussians with short lifespans, achieving a model size of around 50 MB after post-processing. In contrast, other works [6, 11, 23, 28] focus on compressing deformable 3DGS. In this paper, we primarily target Real-Time4DGS [45], by jointly reducing spatio-temporal primitive redundancy and compressing Gaussian attributes.

3 Preliminary

Our framework is designed to be applicable to various Gaussian-based dynamic scene representations, including Real-Time4DGS [45], FreeTimeGS (FTGS) [39], and Spacetime Gaussians (STG) [26]. Despite differences in their temporal parameterizations, these methods share an explicit representation of dynamic scenes using Gaussian primitives.

Real-Time4DGS represents a dynamic scene with a set of 4D Gaussian primitives defined by a 4D mean at spatio-temporal space, an anisotropic 4D covariance, an opacity, and spherical harmonics (SH) coefficients. During rendering, 4D Gaussians are conditioned at timestamp t, yielding 3D position and covariance. In contrast, FreeTimeGS (FTGS) allows Gaussian primitives to be placed freely in space and time, enabling flexible modeling of complex dynamic scenes. Each Gaussian is assigned an explicit motion function based on velocity and a temporal opacity function to control its position and visibility over time. Spacetime Gaussians (STG) models dynamic scenes using Gaussian

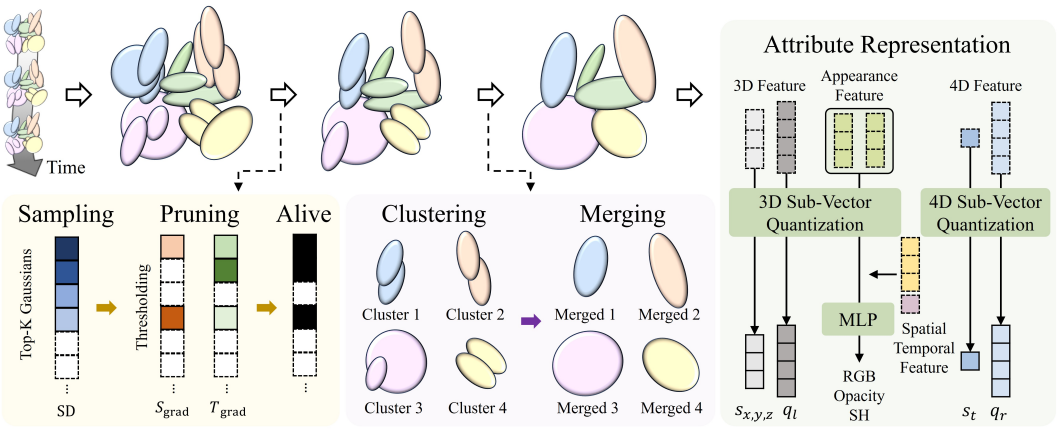


Figure 2: The overall architecture of the proposed *OMG4*.

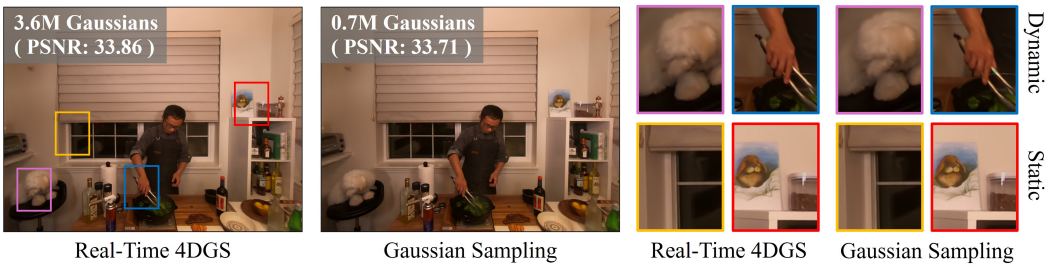


Figure 3: A comparison of rendered images. (Left) Real-Time4DGS [45] with 3.6M Gaussians. (Right) 1K optimization after Gaussian Sampling with 0.7M Gaussians.

primitives with polynomial motion and rotation functions, along with temporal opacity to control their visibility over time. For further details, please refer to the original paper.

Despite their different temporal modeling strategies, all three representations produce time-dependent 3D Gaussian primitives for rendering. This shared Gaussian-based structure enables our framework to operate on individual primitives independently of their temporal parameterizations. We denote the set of Gaussian primitives in a pretrained dynamic scene representation as $\mathcal{P} = (x_i, f_i, a_i)_{i=1}^{N}$, where x_i denotes the reference spatial position, f_i represents the appearance features, and a_i contains the remaining per-primitive attributes, including opacity, scale, rotation, and temporal parameters.

4 Methods

We propose a novel framework for high-fidelity dynamic scene representation with a compact set of Gaussians consisting of Gaussian Sampling (Section 4.1), Gaussian Pruning (Section 4.2), Gaussian Merging (Section 4.3), and Gaussian attributes compression (Section 4.4). An overview of the entire pipeline is provided in Figure 2.

4.1 Gaussian Sampling

Primitive redundancy in dynamic scene representation is heterogeneous across space and time. Although recent studies observed that many Gaussians contribute only marginally to reconstruction quality [22, 50], such redundancy becomes more structured in dynamic scenes due to the distinct spatio-temporal characteristics of static and dynamic regions. Static regions are typically represented by temporally persistent and spatially dispersed Gaussians, while dynamic regions contain temporally localized and spatially dense Gaussians. Thus, spatial-only importance criteria can either over-preserve static background primitives or remove primitives essential for dynamic motion. We therefore propose the Static–Dynamic Score (SD-Score), which evaluates each Gaussian using both spatial rendering sensitivity and temporal consistency. As shown in Figure 3, SD-Score selects a



Space of Score | Unique Gaussians | Optimized Unique Gaussians | Redundant Gaussians

Figure 4: Illustration of Gaussian Pruning. (Left) Space defined by Static and Dynamic score and Gaussian Sampling Boundary. (Middle) A rendered image with unique Gaussians that satisfy both unique static and dynamic thresholds, and one after 1K optimization. (Right) A rendered image with redundant Gaussians that are not included in a unique area.

compact set of salient primitives from a pretrained Real-Time4DGS model [45] while preserving reconstruction quality.

The proposed Static–Dynamic Score (SD-Score) to calculate the importance of each Gaussian is defined as follows:

$$\mathrm{SD}^{(i)} = \mathcal{N}_{\mathrm{minmax}}\left(S^{(i)}_{\mathrm{grad}}\right) \cdot \mathcal{N}_{\mathrm{rank}}\left(T^{(i)}_{\mathrm{grad}}\right), \quad S^{(i)}_{\mathrm{grad}} = \sum_{j=1}^{N} ||\nabla_{u_{i,j}} L_j||_2, \quad T^{(i)}_{\mathrm{grad}} = \sum_{j=1}^{N} \nabla_{\mu_{t_i}} L_j. \tag{1}$$

Here, N is the number of training images (# of input views $\times$ # of frames), L_j is the reconstruction loss for image j, $u_{i,j} \in \mathbb{R}^2$ denotes the projected 2D coordinate of Gaussian i in image j, and $\mu_{t_i} \in \mathbb{R}$ is its temporal coordinate. $\mathcal{N}_{\mathrm{minmax}}(\cdot)$ and $\mathcal{N}_{\mathrm{rank}}(\cdot)$ denote min-max normalization and rank normalization, respectively. Since signed temporal gradients are sensitive to outliers, we apply rank normalization based on their relative ordering, while normalizing both scores to comparable scales before multiplication.

Static Score. The static score $S^{(i)}_{\mathrm{grad}}$ accumulates screen-space gradients over all training images, measuring the spatial rendering sensitivity of each Gaussian with respect to its projected coordinates. In static regions, Gaussians are temporally long-lived yet relatively sparse, so even small positional perturbations can consistently affect the reconstruction loss across many frames, resulting in high static scores. In contrast, Gaussians in dynamic regions are often temporally localized and spatially dense. Consequently, the effect of a single Gaussian's positional change on the overall reconstruction loss is diluted, resulting in lower static scores.

Dynamic Score. The dynamic score $T^{(i)}_{\mathrm{grad}}$ is computed by summing the gradients with respect to the temporal coordinate of each Gaussian, capturing a signed temporal trend in the reconstruction loss. Gaussians with high normalized dynamic scores tend to be important for dynamic regions, where small temporal shifts can substantially affect motion and transient appearance. We accumulate signed time gradients rather than their magnitudes, which acts as an implicit consistency filter: temporally inconsistent Gaussians with frequent gradient sign flips are suppressed through cancellation, whereas Gaussians with coherent temporal trends are better preserved after normalization.

By combining the complementary static and dynamic scores, the SD-Score provides a spatio-temporal importance measure tailored to explicit 4DGS approaches. We sample the Gaussians with high SD values using a sampling ratio of τ_{GS}, forming the sampled set $\mathcal{P}_{\mathrm{GS}}$, and optimize it for T_{GS} iterations.

4.2 Gaussian Pruning

Although Gaussian Sampling selects primitives with high SD-Scores, the sampled set $\mathcal{P}_{GS}$ can still contain redundant Gaussians. In response, we introduce a Gaussian Pruning strategy that further refines the set by eliminating superfluous Gaussians that remain after the first stage.

Figure 4 illustrates the space defined by $\mathcal{N}_{\mathrm{minmax}}(S_{\mathrm{grad}})$ and $\mathcal{N}_{\mathrm{rank}}(T_{\mathrm{grad}})$, where the curve $\mathcal{N}_{\mathrm{rank}}(T_{\mathrm{grad}}) = c/\mathcal{N}_{\mathrm{minmax}}(S_{\mathrm{grad}})$ represents the selection boundary from Gaussian Sampling, with c denoting the SD-Score threshold determined by the sampling ratio τ_{GS}. Within this space, we identify Gaussians that are likely redundant, and our empirical analysis shows that those with both

$S^{(i)}_{\text{grad}}$ and $T^{(i)}_{\text{grad}}$ values being small can be safely filtered out (gray regions in Figure 4). To this end, we apply thresholding to both scores, with each threshold defined as follows:

$$\begin{aligned} \mathcal{P}_{GP} &= \left\{ G_i \in \mathcal{P}_{GS} \;\middle|\; \left(\mathcal{N}(S^{(i)}_{\text{grad}}) \geq \tau_S\right) \vee \left(\mathcal{N}(T^{(i)}_{\text{grad}}) \geq \tau_T\right) \right\}, \\ \tau_S &= Q_p\left(\left\{ \mathcal{N}(S^{(i)}_{\text{grad}}) \right\}_{G_i \in \mathcal{P}_{GS}} \right), \qquad \tau_T = Q_p\left(\left\{ \mathcal{N}(T^{(i)}_{\text{grad}}) \right\}_{G_i \in \mathcal{P}_{GS}} \right), \end{aligned} \tag{2}$$

where τ_S and τ_T are the thresholds for S_{grad} and T_{grad} respectively, and $Q_p(\cdot)$ denotes the p-quantile value computed over the sampled set $\mathcal{P}_{GS}$. As shown in Figure 4, the pruned set, $\mathcal{P}_{GP}$, can still represent the scene with fine details and we optimize $\mathcal{P}_{GP}$ for T_{GP} iterations.

One may question the necessity of employing two distinct stages, since, in principle, sampling and pruning could be integrated into a single process. Nevertheless, our experiments indicate that separating these stages proves to be more effective, as evidenced by the ablation study in Section 5.2.3.

4.3 Gaussian Merging

While Gaussian Sampling and Gaussian Pruning are effective in preserving a compact and salient set of Gaussians, both stages evaluate the importance of each Gaussian individually without considering similarities across Gaussians. To further reduce redundancy, we introduce a Gaussian Merging technique that clusters highly similar Gaussians and fuses them into single representative Gaussians.

We first partition the 4D space-time volume into grid cells and perform clustering only among Gaussians within the same cell. This local grouping reduces the computational cost of pairwise comparison and prevents merging Gaussians that are spatially or temporally mismatched. Within each grid cell, we define the similarity score $S(G_i, G_j)$ between two Gaussians as a sum of spatial proximity and appearance similarity:

$$S(G_i, G_j) = -(||x_i - x_j||_2^2 + \lambda ||f_i - f_j||_2^2), \tag{3}$$

where λ is a fixed balancing weight, $x_i \in \mathbb{R}^3$ is the position, and $f_i \in \mathbb{R}^3$ is the zero-th order spherical-harmonics (RGB) coefficient of G_i. A higher score indicates a greater spatial-appearance similarity. For each Gaussian G_i, we construct a cluster of Gaussians C_i by thresholding the similarity score with τ_{sim}:

$$C_i = \{i\} \cup \{j \in \mathcal{P}_i \mid S(G_i, G_j) \geq \tau_{\text{sim}}\}, \tag{4}$$

where $\mathcal{P}_i$ denotes a set of Gaussian indices within the spatio-temporal grid cell that contain the Gaussian G_i. We then deduplicate identical clusters and remove clusters that are subsets of larger ones, yielding a final set of maximal clusters $\mathcal{C} = \{C_q\}_{q=1}^{N_C}$. Gaussians that do not belong to any non-singleton cluster are kept unchanged.

Given the resulting clusters, we merge the Gaussians within each cluster into a single representative Gaussian using learnable weights for position and appearance. For each cluster C_q, we introduce learnable per-Gaussian weights w_i^x and w_i^f for position and appearance, respectively. During training-time rendering, all Gaussians in C_q are replaced by a single proxy Gaussian:

$$\bar{x}_q = \sum_{i \in C_q} w_i^x x_i, \quad \bar{f}_q = \sum_{i \in C_q} w_i^f f_i, \quad \bar{a}_q = a_{r(C_q)}, \quad r(C_q) = \arg\max_{i \in C_q} w_i^x, \tag{5}$$

where both weights w_i^x, w_i^f are normalized per cluster. The proxy position and appearance are computed as weighted sums of the Gaussians within the cluster, while the remaining attributes $\bar{a}_q$ are taken from the representative Gaussian with the largest position weight. We use the attributes of a representative Gaussian instead of averaging all attributes, because parameters such as scale, rotation, and temporal attributes affect rendering nonlinearly; naively averaging them can yield unstable or poorly representative Gaussians. We optimize both weights w_i^x, w_i^f for T_{GM} iterations and repeat Gaussian Merging M times while progressively increasing the grid size, enabling the model to merge local redundancies first and then consider broader spatio-temporal neighborhoods. The detailed process of Gaussian Merging is provided in the supplementary material.

Table 1: Quantitative results on N3DV [25] at 1352×1014 resolution. Baseline results are sourced from the corresponding papers unless marked with *; * denotes reproduced results. $^\bullet$ denotes results excluding the *Coffee Martini* scene.

Method	PSNR↑	SSIM↑	LPIPS (A)↓	FPS↑	# G↓	Storage (MB)↓
K-Planes [9]	31.63	–	–	0.3	–	311
HexPlane [4]$^\bullet$	31.71	–	0.075	0.2	–	250
Real-Time4DGS [45]*	31.96	0.946	0.051	57	3,397,510	2087
STG [26]	32.05	0.946	0.044	140	–	200
4DGS [41]	31.15	–	0.049	82	–	18.0
MEGA [49]	31.49	–	0.057	77	–	25.05
CSTG [20]	31.69	0.945	0.054	186	–	15.4
GIFStream [23]–L	31.75	0.938	0.051	95	–	10.0
GIFStream [23]*–S	31.25	0.937	0.057	156	–	6.63
4DGS-CC (4DGS) [6]–L	31.54	–	0.062	–	–	3.20
4DGS-CC (4DGS) [6]–S	30.97	–	0.065	–	–	2.30
4DGS-CC (Saro-GS) [6]–L	31.93	–	0.045	–	–	20.9
4DGS-CC (Saro-GS) [6]–S	31.60	–	0.047	–	–	17.0
ADC-GS [11]$^\bullet$–L	31.67	–	0.061	110	–	6.57
ADC-GS [11]$^\bullet$–S	31.41	–	0.066	126	–	4.04
Ours–L	32.03	0.942	0.058	118	228,494	4.71
Ours–M	31.85	0.941	0.060	129	170,044	3.59
Ours–S	31.75	0.939	0.062	133	135,684	2.93
Ours–T	31.54	0.936	0.067	142	97,186	2.18
Ours$^\bullet$–L	32.76	0.949	0.052	123	214,106	4.43
Ours$^\bullet$–T	32.25	0.943	0.063	146	90,930	2.05

4.4 Attribute Compression

Given the compact Gaussian set, we compress attributes by adapting the OMG architecture [22], which is initially designed for static 3DGS, to the 4DGS setting with explicit time conditioning. OMG employs an MLP to extract spatial features and uses three lightweight MLPs to predict opacity, static color, and view-dependent color from the spatial features and position. To handle dynamic scenes, we additionally condition these MLPs on the time coordinate, enabling time-varying opacity and appearance. This compression stage is applied after Gaussian reduction because directly evaluating the MLP-based implicit attribute prediction over the full set of millions of 4D Gaussians introduces substantial memory overhead. Thus, sufficient Gaussian reduction is a necessary prerequisite for tractable attribute compression in explicit 4DGS.

We further extend Sub-Vector Quantization (SVQ) [22], which partitions an input vector into sub-vectors and quantizes each with a small codebook, to compress the attributes of 4D Gaussians. We retain the 4D Gaussian means in full precision for stability, and apply SVQ to the remaining attributes, including the additional rotation quaternion and temporal-axis scales introduced in Real-Time4DGS [45] to model scene motion. However, quantizing both static and dynamic attributes simultaneously leads to unstable optimization, since perturbations to temporal attributes can affect the time-varying geometry and appearance of dynamic primitives, thereby interfering with spatial reconstruction and temporal consistency. To mitigate this, we propose a staged SVQ scheme. We first apply SVQ only to 3D attributes and optimize the compressed representation for T_{3D} iterations, allowing spatial geometry and appearance to stabilize. We then activate SVQ for 4D attributes, enabling temporal attributes to be compressed on top of the stabilized spatial representation. This staged scheme mitigates instability from simultaneous spatial-temporal quantization. Finally, we compress the quantized elements using Huffman encoding [12] followed by LZMA compression [33].

5 Experiments and Results

5.1 Implementation Details

For the reduction stages, we optimize each stage for $1{,}000$ iterations ($T_{GS} = T_{GP} = T_{GM} = 1{,}000$): Gaussian Sampling is followed by Gaussian Pruning at the $1{,}000^{\text{th}}$ iteration, and Gaussian Merging is

Table 2: Quantitative results of *OMG4* on FTGS (FreeTimeGS) [39], evaluated on N3DV [25].

Method	PSNR↑	SSIM↑	LPIPS (A)↓	FPS↑	# G↓	Storage (MB)↓
FTGS	32.80	0.958	0.040	532	500,000	61.04
Ours–L	32.71	0.955	0.044	346	146,891	4.06
Ours–M	32.62	0.954	0.046	354	109,191	3.07
Ours–S	32.45	0.951	0.050	374	80,673	2.32
Ours–T	32.28	0.950	0.051	379	65,706	1.93

Table 3: Quantitative results on SelfCap [43].

Method	PSNR↑	SSIM↑	LPIPS (V)↓	FPS↑	# G↓	Storage (MB)↓
FTGS [39]	26.96	0.908	0.194	241	1,000,000	122
GIFStream [23]*–L	25.83	0.876	0.262	94	–	7.09
GIFStream [23]*–S	25.56	0.866	0.284	104	–	5.52
Ours (FTGS)–L	26.93	0.897	0.224	218	293,402	7.50
Ours (FTGS)–M	26.91	0.897	0.228	235	261,630	6.72
Ours (FTGS)–S	26.88	0.896	0.238	265	201,983	5.24
Ours (FTGS)–T	26.74	0.892	0.254	285	154,155	4.06

applied at the $2{,}000^{\text{th}}$ and $3{,}000^{\text{th}}$ iterations. Attribute compression starts at the $4{,}000^{\text{th}}$ iteration, with MLP training first, 3D SVQ activated at the $9{,}000^{\text{th}}$ iteration, and 4D SVQ at the $10{,}000^{\text{th}}$ iteration. For the medium-size model (*Ours–M*), we set $\tau_{GS} = 0.2$, use a 0.8-quantile pruning threshold, and perform Gaussian Merging twice with a spatial grid scaling factor of 1.2 and temporal grid size of 2.0. We use identical hyperparameters for all scenes within each dataset. Reported storage includes all rendering components, including Gaussian attributes, MLP weights, codebooks, and entropy-coding metadata. More details are provided in the supplementary material.

5.2 Results and Analysis

We first conduct experiments on N3DV [25]. As shown in Table 1 and Figure 5, we effectively reduce the model size of the baseline Real-Time4DGS [45], from 2 GB to around 3 MB, while preserving comparable visual quality (*Ours–M*), corresponding to a 99% reduction in storage. Compared to a recent state-of-the-art method GIFStream [23], *OMG4* achieves 65% reduction in storage (from 10 MB to 3.59 MB) while achieving comparable reconstruction fidelity and slightly higher PSNR (31.75 vs. 31.85). In the low-memory regime, *Ours–T* achieves a better PSNR-storage trade-off than 4DGS-CC [6], with higher PSNR (31.54 vs. 30.97) and smaller storage (2.18 MB vs. 2.30 MB). Beyond compression, *OMG4* improves FPS by 2.26× over the baseline due to its compact representation. We provide rate-distortion curves and BD-rate analyses in the supplementary material.

5.2.1 Application on FreeTimeGS

We further extend *OMG4* to FreeTimeGS [39] to evaluate its generality beyond Real-Time4DGS [45]. Starting from the pretrained FreeTimeGS models[3] (FTGS), *OMG4* reduces Gaussians from 500K to around 147K (Ours–L) and 66K (Ours–T) primitives. As shown in Table 2, *OMG4* reduces the storage of FreeTimeGS by up to 90% while maintaining comparable reconstruction quality, and remains effective even under a strict memory budget of under 2 MB. This result highlights the versatility of *OMG4*, showing that it can be effectively applied to diverse 4D representations. Due to the page limit, results on extending our method to STG [26] are provided in the supplementary material.

5.2.2 Results on Challenging Datasets

Additionally, we conduct experiments under more challenging scenarios, including complex non-rigid motion in SelfCap [43] and long-sequence reconstruction in ENeRF-Outdoor [27]. Table 3 summarizes results on the SelfCap [43] dataset. It consists of six scenes, and we use 60 frames for evaluation following the FreeTimeGS [39] protocol.[4] *OMG4* reduces FTGS from 122 MB to 7.5

[3]We reproduced FreeTimeGS models since the codes and pre-trained models were not publicly available. All experiments were conducted with our reproduced models.

[4]We do not report Real-Time4DGS-based results because it does not provide a reliable pretrained representation under this protocol, making subsequent compression by *OMG4* uninformative.

Table 4: Quantitative results on ENeRF-Outdoor [27]. N denotes the number of models, and F denotes the number of frames in each chunk.

Setting	Method	PSNR↑	SSIM↑	LPIPS (V)↓	FPS↑	#G↓	Storage (MB)↓
$N = 1, F = 1200$	Real-Time4DGS [45]				OOM		
	FTGS [39]	24.95	0.686	0.310	317	4,000,000	488
	Ours (FTGS)	24.76	0.706	0.280	140	1,999,800	52.29
$N = 4, F = 300$	Real-Time4DGS [45]	24.40	0.689	0.293	66	13,121,660	8059
	FTGS [39]	25.04	0.701	0.287	328	4,000,000	488
	GIFStream [23]	23.20	0.662	0.298	44	–	85.85
	Ours (RT4DGS)	24.30	0.660	0.351	111	515,443	11
	Ours (FTGS)	24.70	0.677	0.335	181	1,999,800	53.44
$N = 20, F = 60$	GIFStream [23]	24.23	0.682	0.280	52	–	175



Figure 5: Qualitative results on N3DV dataset [25].

MB with a reasonable fidelity trade-off under aggressive compression. Compared to GIFStream [23], *Ours–T* achieves better reconstruction quality across all metrics with less storage than GIFStream, improving PSNR from 25.56 to 26.74 while reducing storage from 5.52 MB to 4.06 MB. Please refer to the supplementary material for qualitative comparisons.

Table 4 reports results on ENeRF-Outdoor [27] dataset (1200 frames), evaluating long-sequence scenarios. Since Real-Time4DGS [45] and GIFStream [23] cannot be trained on all frames at once due to OOM, we report their results using temporal chunking with four 300-frame models. For a fair comparison, we also evaluate FreeTimeGS [39] and *OMG4* under the same 300-frame chunking protocol. In addition, we follow the original 60-frame chunk setting of GIFStream for comparison. Despite the longer temporal span, *OMG4* achieves visually satisfactory reconstruction quality with a compact memory footprint, demonstrating its robustness under long-sequence scenarios. Notably, the FreeTimeGS-based *OMG4* can also be trained on the full 1200-frame sequence in a single run within an hour, whereas GIFStream requires training 20 chunk-specific models, taking 17 hours in total. This highlights the practical scalability of our method for long dynamic sequences.

5.2.3 Ablation Studies

We present ablation studies on each module of *OMG4* in Table 5. Directly applying Attribute Compression to the unreduced Real-Time4DGS [45] results in out-of-memory failure due to the MLP overhead over millions of primitives, showing that Gaussian reduction is necessary before attribute compression. When applying Gaussian Sampling before attribute compression, we remove 80% of the Gaussians while achieving reconstruction quality comparable to Real-Time4DGS. This may be attributed to a regularization effect, where removing noisy or redundant primitives helps the remaining Gaussians converge more cleanly [1]. Adding Gaussian Pruning further reduces storage to a few megabytes, and Gaussian Merging achieves the minimal number of Gaussians. We also ablate the SVQ schedule inside Attribute Compression. Compared with simultaneous 3D/4D quantization, the proposed staged 3D-to-4D SVQ improves PSNR by 0.31 dB under a similar storage budget, supporting our design of stabilizing spatial attributes before compressing temporal attributes. It may appear possible to perform Gaussian Sampling and Gaussian Pruning simultaneously. However, this variant results in a PSNR drop of 0.15 dB (*GS+GP*). This highlights the importance of the proposed multi-stage pipeline: the intermediate optimization after Gaussian Sampling helps stabilize the

Table 5: Ablation study on each module and SVQ schedule. *GS*, *GP*, *GM*, and *AC* denote Gaussian Sampling, Gaussian Pruning, Gaussian Merging, and Attribute Compression, respectively. *Simul.* denotes simultaneous 3D/4D SVQ, and *Staged* denotes the proposed staged 3D-to-4D SVQ.

Modules					SVQ	PSNR↑	SSIM↑	LPIPS↓	FPS↑	# G↓	Storage (MB)↓
GS+GP	GS	GP	GM	AC	Schedule						
					–	31.96	0.946	0.051	57	3,397,510	2087
				✓	–				OOM		
	✓			✓	Staged	32.02	0.945	0.053	82	679,502	13.25
	✓	✓		✓	Staged	31.94	0.943	0.056	144	241,278	4.83
✓			✓	✓	Staged	31.70	0.941	0.060	121	170,149	3.59
	✓	✓	✓	✓	Simul.	31.54	0.940	0.061	126	170,540	3.59
	✓	✓	✓	✓	Staged	31.85	0.941	0.060	129	170,044	3.59

reduced representation before further pruning. Therefore, our pipeline separates Gaussian Sampling and Gaussian Pruning with an intermediate optimization period. Additional ablation studies are provided in the supplementary material.

6 Conclusion

We introduce *OMG4*, a compact dynamic-scene representation that progressively reduces the number of primitives and compresses the Gaussian attributes. *OMG4* first constructs a compact yet expressive primitive set through Gaussian Sampling, Gaussian Pruning, and Gaussian Merging, which identify salient Gaussians, remove redundant primitives, and fuse correlated ones in space-time. On top of this reduced set, we incorporate implicit appearance encoding and extend Sub-Vector Quantization (SVQ) to 4D representations with a staged 3D-to-4D quantization scheme. Consequently, *OMG4* compresses Real-Time4DGS by three orders of magnitude while maintaining comparable reconstruction fidelity. It also generalizes to other dynamic Gaussian representations, including FreeTimeGS and STG, achieving consistent compression gains with high reconstruction quality. These results show that jointly addressing spatio-temporal primitive redundancy and attribute compression is crucial for compact 4D scene representation, positioning *OMG4* as a practical step toward compact and high-fidelity dynamic scene representations.

Acknowledgments

This research was supported in part by the Ministry of Science and ICT (MSIT) of Korea, under the National Research Foundation (NRF) grant (RS-2024-00337548); the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT) (No.RS-2026-25517417, Development of Compression, Reconstruction, and Rendering Technologies for Free-viewpoint Media); the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00207, Immersive Media Research Laboratory); the IITP grant funded by the Korea government (MSIT): Development of immersive video spatial computing technology for ultra-realistic metaverse services (No.2022-0-00022, RS-2022- II220022); the AI Seoul Tech Research Support Program of the Seoul Future Foundation.

References

[1] Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning, 2024. URL https://arxiv.org/abs/2406.18214.

[2] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting, 2024. URL https://arxiv.org/abs/2404.03613.

[3] Gisle Bjøntegaard. Calculation of average psnr differences between rd-curves. In *VCEG Meeting*, 2001.

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes, 2023. URL https://arxiv.org/abs/2301.09632.

[5] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression, 2024. URL https://arxiv.org/abs/2403.14530.

[6] Zicong Chen, Zhenghao Chen, Wei Jiang, Wei Wang, Lei Liu, and Dong Xu. 4dgs-cc: A contextual coding framework for 4d gaussian splatting data compression, 2025. URL https://arxiv.org/abs/2504.18925.

[7] Pinxuan Dai, Peiquan Zhang, Zheng Dong, Ke Xu, Yifan Peng, Dandan Ding, Yujun Shen, Yin Yang, Xinguo Liu, Rynson WH Lau, et al. 4d gaussian videos with motion layering. *ACM Transactions on Graphics (TOG)*, 44(4):1–14, 2025.

[8] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Light-gaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps, 2024. URL https://arxiv.org/abs/2311.17245.

[9] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance, 2023. URL https://arxiv.org/abs/2301.10241.

[10] Sharath Girish, Tianye Li, Amrita Mazumdar, Abhinav Shrivastava, David Luebke, and Shalini De Mello. Queen: Quantized efficient encoding of dynamic gaussians for streaming free-viewpoint videos, 2024. URL https://arxiv.org/abs/2412.04469.

[11] He Huang, Qi Yang, Mufan Liu, Yiling Xu, and Zhu Li. Adc-gs: Anchor-driven deformable and compressed gaussian splatting for dynamic scene reconstruction, 2025. URL https://arxiv.org/abs/2505.08196.

[12] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[13] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–1, 2024.

[14] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024.

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023. URL https://arxiv.org/abs/2308.04079.

[16] Mustafa Khan, Hamidreza Fazlali, Dhruv Sharma, Tongtong Cao, Dongfeng Bai, Yuan Ren, and Bingbing Liu. Autosplat: Constrained gaussian splatting for autonomous driving scene reconstruction, 2024. URL https://arxiv.org/abs/2407.02598.

[17] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting, 2024. URL https://arxiv.org/abs/2312.00112.

[18] Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting. In *European Conference on Computer Vision*, pages 127–143. Springer, 2024.

[19] Dong In Lee, Hyeongcheol Park, Jiyoung Seo, Eunbyung Park, Hyunje Park, Ha Dam Baek, Sangheon Shin, Sangmin Kim, and Sangpil Kim. Editsplat: Multi-view fusion and attention-guided optimization for view-consistent 3d scene editing with 3d gaussian splatting, 2025. URL https://arxiv.org/abs/2412.11520.

[20] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian splatting for static and dynamic radiance fields, 2024. URL https://arxiv.org/abs/2408.03822.

[21] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field, 2024. URL https://arxiv.org/abs/2311.13681.

[22] Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized minimal 3d gaussian splatting, 2025. URL https://arxiv.org/abs/2503.16924.

[23] Hao Li, Sicheng Li, Xiang Gao, Abudouaihati Batuer, Lu Yu, and Yiyi Liao. Gifstream: 4d gaussian-based immersive video with feature stream, 2025. URL https://arxiv.org/abs/2505.07539.

[24] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video, 2022. URL https://arxiv.org/abs/2103.02597.

[25] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5521–5531, 2022.

[26] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis, 2024. URL https://arxiv.org/abs/2312.16812.

[27] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia Conference Proceedings*, 2022.

[28] Mufan Liu, Qi Yang, He Huang, Wenjie Huang, Zhenlong Yuan, Zhu Li, and Yiling Xu. Light4gs: Lightweight compact 4d gaussian splatting generation via context model, 2025. URL https://arxiv.org/abs/2503.13948.

[29] Zhening Liu, Yingdong Hu, Xinjie Zhang, Jiawei Shao, Zehong Lin, and Jun Zhang. Dynamics-aware gaussian splatting streaming towards fast on-the-fly training for 4d reconstruction. *Training*, 101:102, 2024.

[30] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering, 2023. URL https://arxiv.org/abs/2312.00109.

[31] Seungjun Oh, Younggeun Lee, Hyejin Jeon, and Eunbyung Park. Hybrid 3d-4d gaussian splatting for fast dynamic scene representation, 2025. URL https://arxiv.org/abs/2505.13215.

[32] Zipeng Pan, Yuan Zhang, and Tao Lin. Telegs: End-to-end monocular gaussian head for immersive telepresence. In *Proceedings of the 3rd Workshop on Emerging Multimedia Systems*, pages 37–42, 2025.

[33] Igor Pavlov. Lzma compression algorithm. *https://www.7-zip.org/sdk.html*.

[34] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes, 2020. URL https://arxiv.org/abs/2011.13961.

[35] Seungjoo Shin, Jaesik Park, and Sunghyun Cho. Locality-aware gaussian compression for fast and high-quality rendering, 2025. URL https://arxiv.org/abs/2501.05757.

[36] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos, 2024. URL https://arxiv.org/abs/2403.01444.

[37] Yecong Wan, Mingwen Shao, Yuanshuo Cheng, and Wangmeng Zuo. S2gaussian: Sparse-view super-resolution 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 711–721, 2025.

[38] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis, 2023. URL https://arxiv.org/abs/2212.00190.

[39] Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Freetimegs: Free gaussian primitives at anytime and anywhere for dynamic scene reconstruction, 2025. URL https://arxiv.org/abs/2506.05348.

[40] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting, 2025. URL https://arxiv.org/abs/2403.11868.

[41] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering, 2024. URL https://arxiv.org/abs/2310.08528.

[42] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, et al. Vr-nerf: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023.

[43] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4k4d: Real-time 4d view synthesis at 4k resolution. 2023.

[44] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024.

[45] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting, 2024. URL https://arxiv.org/abs/2310.10642.

[46] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction, 2023. URL https://arxiv.org/abs/2309.13101.

[47] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024.

[48] Yuheng Yuan, Qiuhong Shen, Xingyi Yang, and Xinchao Wang. 1000+ fps 4d gaussian splatting for dynamic scene rendering, 2025. URL https://arxiv.org/abs/2503.16422.

[49] Xinjie Zhang, Zhening Liu, Yifan Zhang, Xingtong Ge, Dailan He, Tongda Xu, Yan Wang, Zehong Lin, Shuicheng Yan, and Jun Zhang. Mega: Memory-efficient 4d gaussian splatting for dynamic scenes, 2025. URL https://arxiv.org/abs/2410.13613.

[50] Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. Lp-3dgs: Learning to prune 3d gaussian splatting, 2024. URL https://arxiv.org/abs/2405.18784.

[51] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes, 2024. URL https://arxiv.org/abs/2312.07920.

Optimized Minimal 4D Gaussian Splatting for Efficient Dynamic Scene Representation: Supplementary Material

A Additional Experimental Results

A.1 Rate-Distortion Curves and BD-Rate Analysis

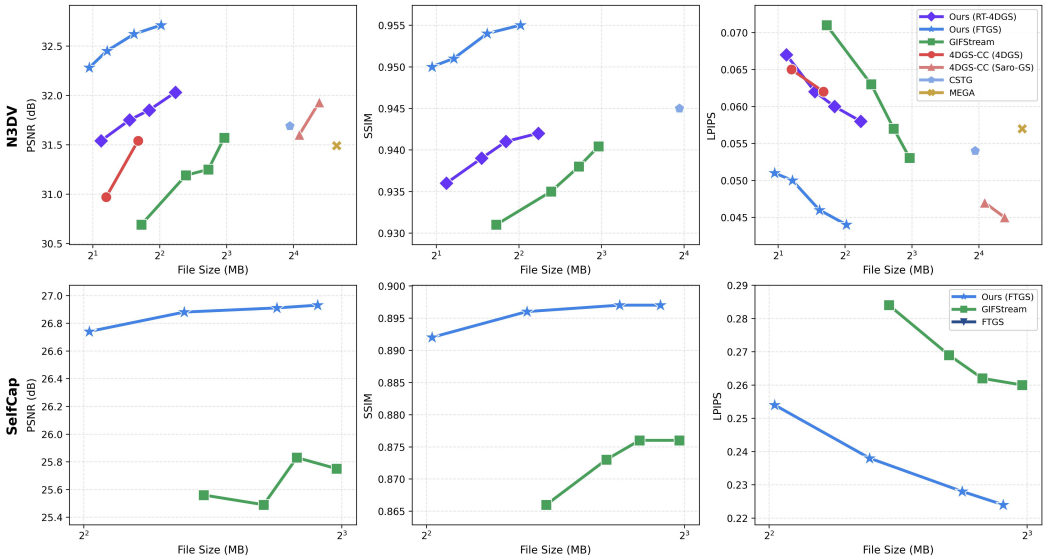


Figure 6: Rate-distortion curves on N3DV [25] (Top) and SelfCap [43] (Bottom).

We provide rate-distortion curves in Figure 6, using storage size as the rate and PSNR, SSIM, and LPIPS as reconstruction quality metrics. On N3DV, the RT-4DGS-based *OMG4* consistently achieves a favorable rate-distortion trade-off. Compared to GIFStream [23], *OMG4* attains higher PSNR and SSIM with substantially smaller storage. In the low-memory regime, it also provides a better PSNR-storage trade-off than 4DGS-CC (4DGS) [6]. The FTGS-based *OMG4* further improves the overall RD curve, demonstrating that the proposed compression pipeline can benefit from stronger dynamic Gaussian backbones.

We compute PSNR-based BD-rate using the Bjøntegaard delta metric [3], where storage size is used as the rate. BD-rate is computed over the overlapping PSNR range between each baseline and *OMG4*; negative values indicate storage savings at the same PSNR. For the RT-4DGS-based *OMG4*, the BD-rate results show substantial storage savings on N3DV: -71.65% compared to GIFStream, and -83.79% compared to 4DGS-CC (Saro-GS). When the PSNR ranges do not overlap but storage ranges overlap, we report BD-PSNR instead. For example, the FTGS-based *OMG4* improves BD-PSNR by $+1.30$ dB over 4DGS-CC (4DGS) on N3DV.

On SelfCap, the FTGS-based *OMG4* consistently outperforms GIFStream across the RD curves. Since the PSNR ranges of GIFStream and *OMG4* do not overlap, PSNR-based BD-rate is not available. Instead, over the overlapping storage range, *OMG4* improves BD-PSNR by $+1.30$ dB on average, further confirming its robustness on complex non-rigid motions.

A.2 More Ablation Studies

Ablation on Signed Temporal Gradients. We further analyze the design of the Dynamic Score in the proposed SD-Score. In the main method, the temporal component is computed by accumulating signed gradients with respect to the temporal coordinate:

$$T_{\text{grad}}^{(i)} = \sum_{j=1}^{N} \nabla_{\mu_{t_i}} L_j. \tag{6}$$

Table 6: PSNR-based BD-rate and BD-PSNR analysis. Storage size is used as the rate. Negative BD-rate indicates storage savings of *OMG4* at the same PSNR, while positive BD-PSNR indicates higher PSNR at the same storage.

Dataset / Variant	Baseline	BD-Rate / PSNR (%) ↓	BD-PSNR (dB) ↑
N3DV / Ours (RT-4DGS)	GIFStream [23]	-71.65	–
N3DV / Ours (RT-4DGS)	4DGS-CC (4DGS) [6]	–	$+0.44$
N3DV / Ours (RT-4DGS)	4DGS-CC (Saro-GS) [6]	-83.79	–
N3DV / Ours (FTGS)	4DGS-CC (4DGS) [6]	–	$+1.30$
SelfCap / Ours (FTGS)	GIFStream [23]	–	$+1.30$

Table 7: Ablation on the Dynamic Score of SD-Score.

Dynamic Score	PSNR↑	SSIM↑	LPIPS↓
Absolute acc.	31.65	0.941	0.060
Signed acc. (Ours)	31.85	0.941	0.060

An alternative design is to accumulate gradient magnitudes:

$$T_{\text{abs}}^{(i)} = \sum_{j=1}^{N} \left| \nabla_{\mu_{t_i}} L_j \right|. \tag{7}$$

This variant measures the overall magnitude of temporal sensitivity, but ignores the directional consistency of temporal gradients across views and frames. As a result, Gaussians whose temporal gradients frequently change signs can still receive high scores, even though such sign changes may indicate unstable or flickering temporal behavior. In contrast, signed accumulation acts as an implicit temporal consistency filter. If a Gaussian exhibits inconsistent temporal gradients across views or frames, the positive and negative gradients are partially canceled before normalization, reducing its Dynamic Score. On the other hand, Gaussians with coherent temporal trends are better preserved. This behavior is desirable for aggressive compression, where the selected primitives should not only be sensitive to temporal changes but also provide stable temporal contributions. As shown in Table 7, replacing the proposed signed temporal accumulation with absolute-gradient accumulation (*Absolute acc.*) decreases PSNR from 31.85 to 31.65. This result supports our design choice: the signed temporal gradient provides a more effective ranking signal than magnitude-only temporal sensitivity for selecting dynamic primitives.

Ablation on Gaussian Merging Design. We further analyze the design choice in Gaussian Merging. In the main method, we learn merging weights only for position and appearance, while the remaining attributes, such as scale, rotation, and temporal attributes, are inherited from the representative Gaussian. This design is motivated by the fact that position and RGB-like appearance are naturally defined in Euclidean spaces, making weighted combinations stable and interpretable. In contrast, scale, rotation, and temporal attributes affect rendering in a nonlinear manner, and naively blending them can produce poorly representative proxy Gaussians. To validate this design, we compare two variants. First, we additionally learn weights for covariance-related parameters, including scale and rotation. Second, we further learn weights for temporal attributes. As shown in Table 8, both variants slightly degrade reconstruction quality compared to the proposed design. This suggests that directly blending covariance-related or temporal attributes does not necessarily produce a better representative Gaussian, even though it increases the flexibility of the merging operation. Therefore, we use learnable weighted sums only for position and appearance, and inherit the remaining attributes from the representative Gaussian.

Hyperparameter Analysis for Gaussian Reduction. We further conduct ablation studies on the proposed method, evaluated on *cook spinach* and *sear steak* data of N3DV dataset [25]. We compare each hyperparameter of the Gaussian removal stage. Table 9 and Table 10 show the comparison on different pruning ratio p and the number of Gaussian Merging implementation, respectively. As expected for Gaussian-based models, the reconstruction quality increases monotonically with the number of Gaussians. We experimentally set these hyperparameters, considering the storage and image fidelity tradeoff.

Table 8: Ablation on Gaussian Merging design. The proposed method learns merging weights only for position and appearance and uses position-appearance similarity for clustering.

Variant	PSNR↑	SSIM↑	LPIPS↓
Ours	31.85	0.941	0.060
+ Covariance-weighted merging	31.72	0.941	0.060
+ Temporal-weighted merging	31.75	0.941	0.060

Table 9: Ablation study on the quantile level p of Gaussian Pruning.

Setting	PSNR↑	SSIM↑	LPIPS↓	# G↓	Storage (MB)↓
$p = 0.6$	33.56	0.956	0.046	231,984	4.78
$p = 0.7$	33.55	0.956	0.046	192,789	4.02
$p = 0.8$	33.45	0.955	0.048	143,880	3.09
$p = 0.9$	32.94	0.949	0.057	82,605	1.89

A.3 Application on STG

We further apply *OMG4* to STG [26], to validate the generality of the proposed method. STG extends 3D Gaussian Splatting [15] by incorporating temporal components to model dynamic scenes. In STG, the parameters of each Gaussian vary over time to capture object motions or appearance changes in the scene. Specifically, the spatial position of each Gaussian is modeled using a polynomial motion trajectory that predicts time-dependent positional displacements. For opacity and rotation, STG leverages a radial basis function and a polynomial function to capture temporal variations, respectively. In addition, to model appearance, spherical harmonics are replaced with neural features.

As shown in Table 11, *OMG4* can achieve comparable reconstruction quality to CSTG [20], which is a compression technique designed for STG, while reducing the memory footprint by 27%. Moreover, when using approximately 1.6 MB more memory than CSTG, *OMG4* further improves the visual quality, demonstrating the scalability of our method. Combined with the results on Real-Time4DGS [45] and FreeTimeGS [39], these results verify that *OMG4* generalizes well across different 4D Gaussian representation frameworks.

A.4 Additional Results on N3DV

We additionally report N3DV results at 1024×768 resolution in Table 12.[5] *OMG4* achieves a strong rate-distortion trade-off, reducing 4DGS-1K [48] from 49.50 MB to 3.63 MB while improving PSNR (31.87 vs. 32.13). It also outperforms Light4GS–S [28] with less storage (31.48 vs. 32.13, 3.77 MB vs. 3.63 MB), confirming consistent compression gains.

B Analysis on Method

B.1 SD Score

In this section, we analyze the proposed SD score in Gaussian Sampling stage. Gaussian Sampling aims to extract a subset of Gaussians that can faithfully represent both static and dynamic regions. Previous work [31] has attempted to leverage the variance of time distribution, Σ_t, as a cue to detect separate static and dynamic regions. Gaussians located at the regions with various motions tend to have relatively smaller values of Σ_t compared to those lying on the stationary area, as rapid changes in appearance or geometry restrict each primitive to have a short lifespan, being temporally localized. However, using Σ_t alone to select a subset with a limited budget can be biased toward extremes because sampling dynamic Gaussians by taking the bottom Σ_t subset focuses on a few high-velocity regions (e.g., moving hands in Figure 7), while neglecting other dynamic structures such as secondary body motion. This is because Σ_t is agnostic to the visual contribution, oversampling transients that are temporally sharp. On the other hand, our SD Score addresses this by combining gradient-based contribution with temporal support. Under the same sampling budget, it can distribute the

[5] We report this additional comparison at 1024×768 because some baselines only provide results at this resolution.

Table 10: Ablation study on the number of Gaussian Merging rounds M.

Setting	PSNR↑	SSIM↑	LPIPS↓	# G↓	Storage (MB)↓
$M = 1$	33.49	0.955	0.047	164,788	3.49
$M = 2$	33.45	0.955	0.048	143,880	3.09
$M = 3$	33.43	0.954	0.049	134,637	2.91
$M = 4$	33.36	0.954	0.049	129,664	2.81

Table 11: Quantitative results of applying *OMG4* to STG [26] on N3DV [25].

Method	PSNR↑	SSIM↑	LPIPS (A)↓	# G↓	Storage (MB)↓
STG [26]	32.01	0.948	0.046	1,289,577	172
CSTG [20]	31.69	0.945	0.054	–	15.4
Ours–L	31.90	0.947	0.046	418,085	16.99
Ours–S	31.72	0.945	0.050	264,401	11.17

Table 12: Quantitative results on N3DV [25] at 1024×768 resolution. Baseline results are sourced from the corresponding papers unless marked with *; * denotes reproduced results. $^\bullet$ denotes results excluding the *Coffee Martini* scene.

Method	PSNR↑	SSIM↑	LPIPS (A)↓	FPS↑	# G↓	Storage (MB)↓
Real-Time4DGS [45]*	32.21	0.950	0.040	96	2,770,350	1701
4DGS-1K [48]	31.87	0.944	0.053	805	666,632	49.50
Light4GS [28]$^\bullet$–L	31.69	-	0.053	37	-	5.46
Light4GS [28]$^\bullet$–S	31.48	-	0.064	40	-	3.77
Ours–L	32.21	0.948	0.044	162	231,537	4.77
Ours–M	32.13	0.946	0.045	178	171,965	3.63
Ours–S	32.08	0.945	0.048	185	136,679	2.95
Ours–T	31.79	0.942	0.052	193	98,217	2.20
Ours$^\bullet$–L	32.95	0.953	0.040	166	217,401	4.50
Ours$^\bullet$–T	32.55	0.947	0.048	198	92,033	2.08

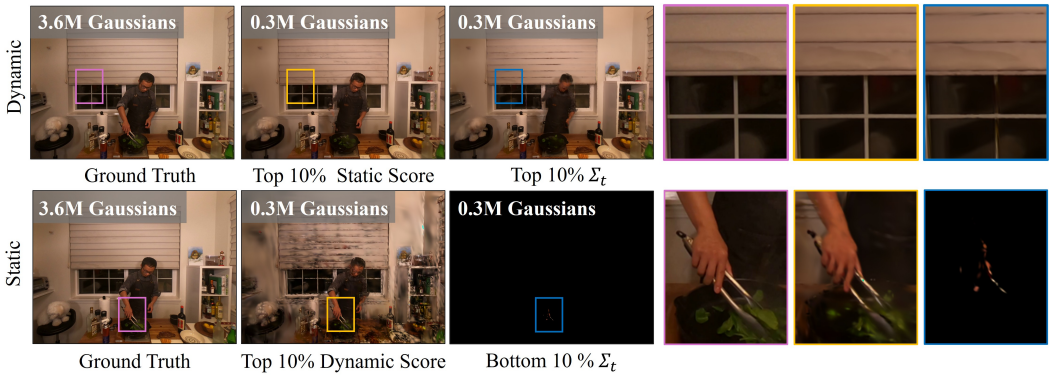


Figure 7: Effect of the sampling criterion under a fixed sampling budget. (Left) Rendered images with the full model. (Middle): Our SD Score selection using top 10% Static score and Dynamic Score. (Right) Σ_t-based sampling with the same sampling ratio.

budget across multiple dynamic regions while keeping enough static structures, as shown in Figure 7, producing better perceptual fidelity with the same number of Gaussians.

Moreover, some online 4D reconstruction approaches [7, 29] adopts 2D optical flow to distinguish dynamic and static Gaussians using 2D optical flow. To further examine whether such approach can benefit our pipeline, we adopt the optical-flow–based dynamic region estimation proposed in [29] and use it as a Dynamic Score. While streaming-based methods typically compute optical flow of

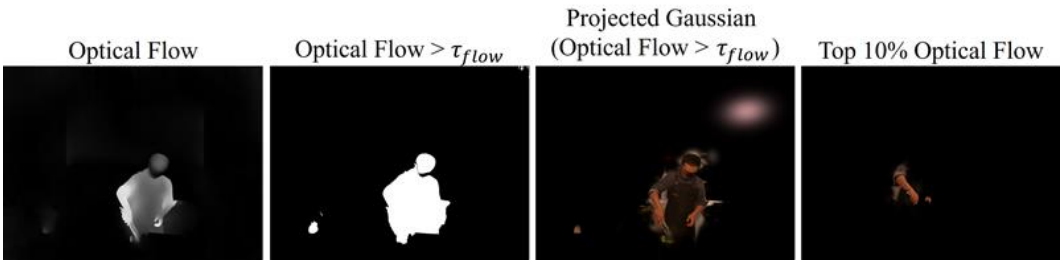


Figure 8: Visualization of 2D optical flow-based Dynamic score. (Left) Predicted 2D optical flow. (Middle-Left): Optical flow thresholded with τ_{flow}. (Middle-Right) Rendered image with the Gaussians projected on high-optical flow region. (Right): Rendered image with the Gaussians of top 10% Dynamic score.

Table 13: Ablation on replacing the proposed Dynamic Score with optical flow.

Method	PSNR↑	SSIM↑	LPIPS↓	# G↓	Storage (MB)↓
Optical flow	27.19	0.840	0.156	150,889	3.20
Temporal gradient (Ours)	31.75	0.939	0.062	135,684	2.93

each streamed frame, our method requires evaluating each Gaussian's contribution to the entire scene globally. To this end, we accumulate optical-flow magnitudes across all frames. We then project each Gaussian onto the image plane and accumulate the flow magnitudes of pixels that fall within the corresponding 2D dynamic regions, thereby assigning an optical flow magnitude to each 4D Gaussian. If the accumulated optical flow exceeds a threshold τ_{flow}, we mark such Gaussians as dynamic Gaussians.

The segmented results are provided in Figure 8 and quantitative results on the N3DV dataset are delivered in Table 13. The optical flow-based variant performs noticeably worse than our temporal gradient-based Dynamic Score in terms of both reconstruction quality and compression efficiency. We assume this is because optical flow is highly sensitive to frame-to-frame pixel changes and thus primarily highlights a few extremely fast-moving regions, rather than Gaussians that are essential for maintaining the global scene structure. As illustrated in Figure 8, the top 10% Gaussians ranked by optical flow predominantly capture highly dynamic parts, but fail to select representative dynamic Gaussians necessary for preserving the overall scene, similar to using Σ_t [31]. In other words, optical flow mainly measures how much a Gaussian move and does not guarantee sufficient coverage of the entire dynamic region under a tight sampling budget. In contrast, our temporal gradient-based Dynamic Score directly measures how much each Gaussian contributes to the reconstruction loss over time, leading to far more stable and reliable selections.

B.2 Gaussian Merging

Algorithm 1 describes a detailed process of Gaussian Merging. As mentioned earlier, our Gaussian Merging performs correlation-considered Gaussian removal based on the locality of Gaussians, while Gaussian Sampling and Gaussian Pruning rather aim to identify the salience or contribution of a single Gaussian. Also, computing proxy Gaussians ($\mathcal{P}_{\text{proxy}}$ in algorithm 1) is differentiable, so the entire pipeline, including Gaussian Merging, can be trained end-to-end. It is worth to mention that once the weights (w_i^x and w_i^f in algorithm 1) are learned, Gaussians in each clusters are permanently merged via weighted sum; therefore, relevant training-time overhead disappears during the rest of the process (*i.e.,* Attribute Compression). At test time, since the Gaussians are already merged, the merging operation incurs no extra cost and does not affect FPS.

B.3 Training Time Analysis

Table 14 reports the optimization time of *OMG4* applied to Real-Time4DGS [45] on the N3DV dataset [25], measured with an RTX 4090 GPU. We train *OMG4* for a total of 11,000 iterations, and, before optimization, we perform a preprocessing step that computes view-space and temporal

Algorithm 1 Gaussian Merging

Require: Gaussian set $\mathcal{P}_{GP}$; clusters $\mathcal{C} = \{C_q\}_{q=1}^{N_C}$; training steps T_M

Initialize per-member logits $\ell_i^x \leftarrow 0,\ \ell_i^f \leftarrow 0$ for all i
for $t = 1$ **to** T_M **do**
  for all clusters C_q **do**
    $w_i^x \leftarrow \sigma(\ell_i^x) \big/ \sum_{j \in C_q} \sigma(\ell_j^x) \quad$ for $i \in C_q$ ▷ Position weights
    $w_i^f \leftarrow \sigma(\ell_i^f) \big/ \sum_{j \in C_q} \sigma(\ell_j^f) \quad$ for $i \in C_q$ ▷ Appearance weights
    $\bar{x}_q \leftarrow \sum_{i \in C_q} w_i^x \, x_i; \quad \bar{f}_q \leftarrow \sum_{i \in C_q} w_i^f \, f_i$
    $\bar{a}_q \leftarrow a_{r(C_q)}, \quad r(C_q) = \arg\max_{i \in C_q} w_i^x$
  end for
  $\mathcal{P}_{\text{proxy}} \leftarrow \{(\bar{x}_q, \bar{f}_q, \bar{a}_q)\}_{q=1}^{N_C} \cup \{(x_i, f_i, a_i) : i \notin \bigcup_q C_q\}$
  Render $\mathcal{P}_{\text{proxy}}$, compute loss $\mathcal{L}$, and update $\{\ell_i^x, \ell_i^f\}$ by backprop
end for
return pruned set $\mathcal{P}_{\text{proxy}}$

Table 14: Training time analysis.

Method	Sub-module	# Iterations	Time (min)	Time / iter (sec)
Pretraining	–	30,000	155.05	0.310
OMG4	–	11,000	45.17	0.246
*OMG4**	–	11,000	28.83	0.157
Gradient Computing	–	–	16.35	–
Gaussian Sampling	–	1,000	2.47	0.148
Gaussian Pruning	–	1,000	2.41	0.144
Gaussian Merging 1	Cluster calculation	–	0.83	–
Gaussian Merging 1	Optimization	1,000	2.57	0.154
Gaussian Merging 2	Cluster calculation	–	0.59	–
Gaussian Merging 2	Optimization	1,000	2.44	0.146
Attribute Compression	MLP	5,000	12.14	0.146
Attribute Compression	MLP + 3D SVQ	1,000	2.57	0.154
Attribute Compression	MLP + 3D SVQ + 4D SVQ	1,000	2.83	0.170

gradients. The row "OMG4" therefore includes both this preprocessing and the subsequent optimization, while "OMG4*" excludes the gradient computation and reports only the pure optimization time. Thanks to our effective reduction of the Gaussian count, the per-iteration optimization time decreases from 0.31 s/iter for the original pretraining stage to 0.157 s/iter for OMG4*, i.e., nearly a $2\times$ speedup. The lower part of the table further decomposes the wall-clock time of each module. "Gradient Computing" corresponds to the one-time cost of preprocessing.

Both "Gaussian Sampling" and "Gaussian Pruning" run for 1,000 iterations. "Gaussian Pruning" is applied after "Gaussian Sampling" has already reduced the number of points, which explains its slightly lower per-iteration time. For "Gaussian Merging", we separately list the time for cluster computation ("Calc. Clusters") and the subsequent optimization. During merging, Gaussians are temporarily combined for rendering, which introduces a small overhead of about 0.01 second per iteration; in the second merging stage, the number of Gaussians is already reduced, so both clustering and optimization are faster than in the first stage.

In the "Attribute Compression" block, we show that using a small MLP for attribute encoding has a negligible impact on runtime and that the additional stages with SVQ-3D and SVQ-4D introduce only a modest increase in per-iteration time. After optimization is done, decoding the compressed model takes approximately 8 seconds, which is a one-time cost. Overall, these results demonstrate that our pipeline substantially reduces optimization time per iteration while keeping each compression and merging component computationally lightweight.

C Additional Implementation Details

In this section, we describe the implementation details and experimental settings in detail.

C.1 Real-Time4DGS setup

As mentioned earlier, we follow the Sub-Vector Quantization (SVQ) configuration of OMG [22], where the codebook sizes for scale b_s, rotation b_r, and encoded appearance feature b_f are set to 2^9, 2^{13}, and 2^{10}, respectively. Since Real-Time4DGS [45] introduces additional attributes, namely s_t (the scale along the temporal axis) and q^l (an additional quaternion used to define the rotation matrix), we also apply SVQ to these attributes, using codebook sizes of 2^9 and 2^{13}, respectively. We use this configuration for all datasets and all variants. We use τ_{GS} as 0.2, 0.3, 0.2 and quantile p as 0.7, 0.9, 0.9 for our large, small and tiny variants, respectively. For the ENeRF-Outdoor [27] dataset, we set $\tau_{GS} = 0.2$ and use the 0.9-quantile.

C.2 Free-TimeGS setup

Similar to the Real-Time4DGS [45] setting, we also apply sub-vector quantization to the additional attributes introduced by FreeTimeGS, namely velocity and duration. We use the same codebook sizes for both model scales, setting the codebook size of both velocity and duration to 2^{10}. For the N3DV [25] dataset, we set τ_{GS} to 0.5, 0.7, 0.5, 0.4 and quantile p as 0.6, 0.8, 0.8, 0.8 for our large, medium, small, and tiny variants and set τ_{GS} to 0.4, 0.4, 0.3, 0.3 and quantile p as 0.2, 0.4, 0.4, 0.6 for our large, medium, small, and tiny variants in SelfCap [43] dataset. For the ENeRF-Outdoor [27] datasets, we set $\tau_{GS} = 0.5$ and use the 0.2-quantile for Gaussian Pruning.

As shown in the ablation studies provided earlier, we found that the reconstruction quality generally improves as more Gaussians are preserved. This scalability allows our method to flexibly control the rate-distortion trade-off, making it adaptable to different memory budgets and practical requirements.

D Qualitative Comparisons

We provide visual comparisons of SelfCap [43] and ENeRF-Outdoor [27] dataset in Figure 9 and Figure 10.

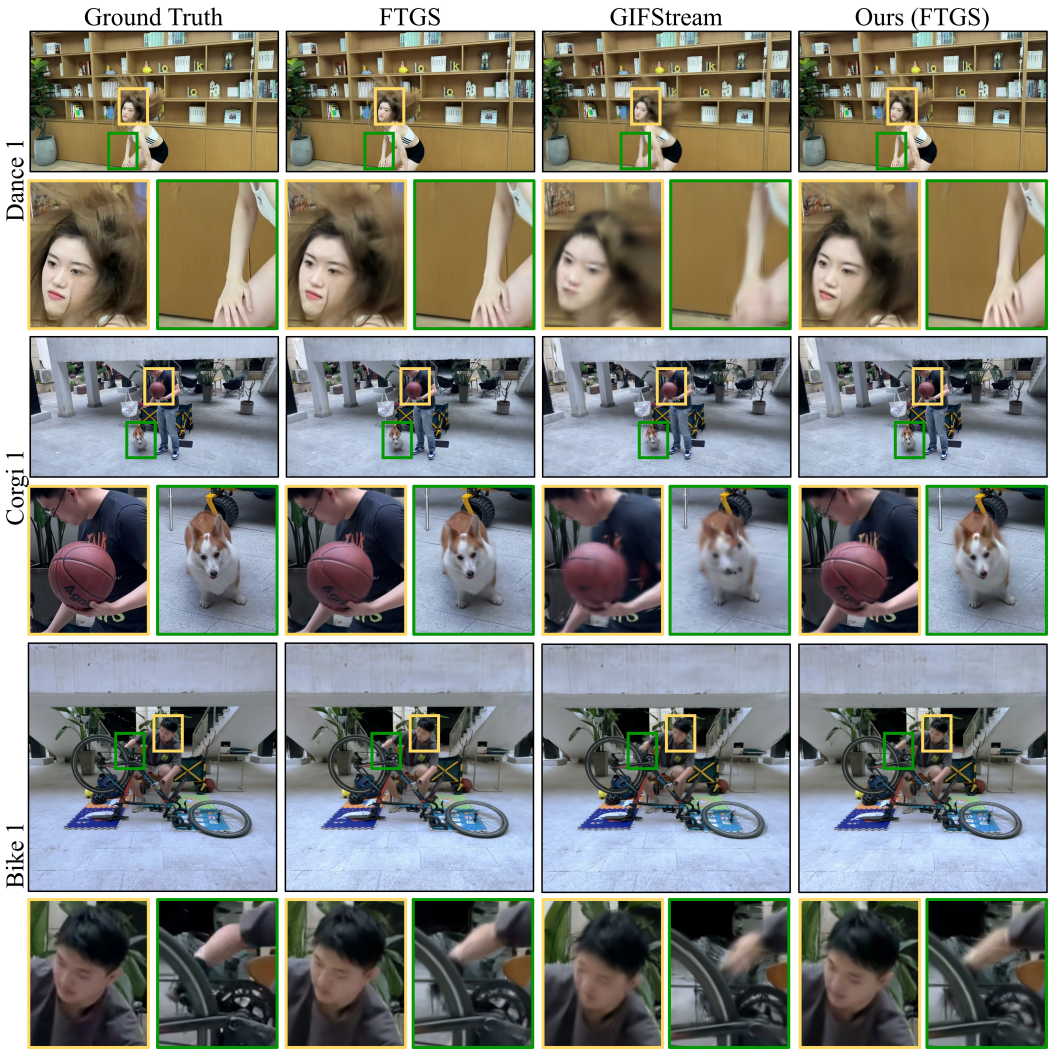


Figure 9: Qualitative results on SelfCap dataset [43].

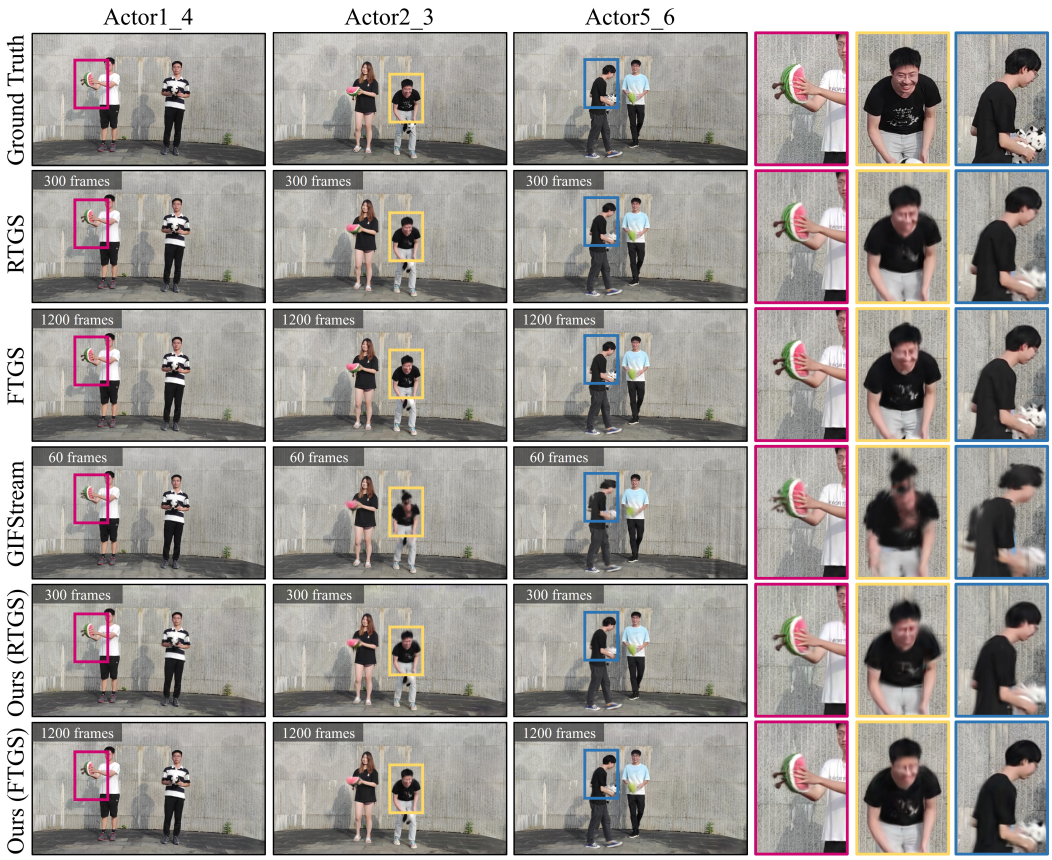


Figure 10: Qualitative results on ENeRF-Outdoor dataset [27].

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 abstract and introduction clearly state the main contribution of the paper, and the claims are supported by the experimental results.

   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: [No]

   Justification: We do not include a separate limitations section in the current submission, although our claims are restricted to the evaluated datasets, baselines, and compression settings.

   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 include theoretical results.

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 provides experimental settings in detail.

   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:[Yes]

Justification: The paper uses publicly accessible data and will release the code.

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 clearly specifies all training and test details, and will release the code for better reproduction.

   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 provides quantitative comparisons under standard evaluation settings, but does not report error bars or statistical significance analyses.

   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 provides computational analysis with the type of GPU.

   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 research follows the NeurIPS Code of Ethics.

   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: [N/A]

    Justification: There is no societal impact of the work performed in the paper.

    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 pose such risks.

    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 original works that produced any dataset or models.

    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: [N/A]

    Justification: The paper does not release any new assets.

    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 involv any crowdsourcing research.

    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 paper does not involve any research of human subjects.

    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 development in this research does not involve LLMs as any import, original or non-standard components.

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.