PGMS: Pyramidal Gaussian Mixture Splatting for 3DGS Compression

Xiaoge Zhang
The University of Western Australia
xiaoge.zhang@uwa.edu.au

Zijie Wu*
The University of Western Australia
zijie.wu@uwa.edu.au

Mingtao Feng
Xidian University
mingtao.feng@xidian.edu.cn

Saeed Anwar
The University of Western Australia
saeed.anwar@uwa.edu.au

Ajmal Mian
The University of Western Australia
ajmal.mian@uwa.edu.au



Figure 1: Our method can preserve high-fidelity geometry and rich appearance, while achieving 74× and 9× compression relative to 3DGS and Scaffold-GS, respectively.

Abstract

Compressing 3D Gaussian Splatting (3DGS) is essential for reducing the storage costs of its dense, high-dimensional representation while preserving high-quality rendering. Existing methods fit compact representations to dense Gaussian sets, potentially allocating representational capacity to primitives with negligible rendering contributions and retaining their influence in the compressed representation. We propose **Pyramidal Gaussian Mixture Splatting** (PGMS), a rate-controllable pyramidal representation that aligns representation capacity with rendering fidelity through structured Gaussian aggregation. PGMS first constructs a hierarchical mixture primitive pyramid from dense Gaussians, which jointly reduce primitive-level redundancy and attribute redundancy within each Gaussian. We further introduce a rate-adaptive capacity allocator conditioned on local Gaussian statistics and a target parameter reduction ratio, enabling scene-adaptive allocation of geometric and appearance capacity across pyramid levels to improve rendering fidelity under a prescribed parameter budget. Finally, a compositing-aware parameter solver initializes each mixture primitive by aggregating dense Gaussians according to their visible

*Corresponding author.

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

color and occlusion contributions, preserving rendering-relevant information after reduction. Experiments on Mip-NeRF 360, Tanks & Temples, and Deep Blending demonstrate that PGMS achieves better rate–distortion trade-offs than prior 3DGS compaction and compression methods across vanilla 3DGS, Scaffold-GS, and MiniSplatting backbones.

1 Introduction

3D Gaussian Splatting (3DGS) [23] has emerged as a promising 3D representation due to its photorealistic quality and real-time rendering speed. It represents a scene by learning a set of anisotropic Gaussian primitives that jointly encode geometry and appearance. Owing to its simple yet expressive formulation, 3DGS has been widely adopted in applications such as digital humans [54, 40], autonomous driving [21, 57], and medical imaging [5, 22]. However, the large number of Gaussians and their associated attributes impose substantial challenges for storage and transmission.

To alleviate the storage burden of large-scale Gaussian data, recent works on 3DGS compression mainly follow two directions. The first adopts an attribute compression paradigm, where geometric and appearance attributes are quantized, with bit allocation and context modeling guided by either hand-crafted sensitivity [45] or learned priors [30, 29, 6]. While effective in reducing storage, these methods focus on quantization and coding, lacking explicit modeling of structural redundancy within and across Gaussians. Consequently, under low-bitrate settings, they may fail to capture the varying importance of Gaussians, leading to the loss of critical geometric and appearance details. The second direction reduces redundancy at the representation level. Instead of storing many independent Gaussians locally, Scaffold-GS [33] uses a small set of anchors to generate multiple Gaussians, while other methods reparameterize attributes into compact forms such as factorized tensors [44], feature planes [49, 28], grid-based fields [27], or locality-aware representations [43]. By removing redundancy in the representation itself, these approaches often achieve lower storage with comparable rendering quality. Different from both, compaction methods [19, 47, 55, 46] directly prune primitives with negligible rendering impact, enabling lightweight 3DGS. However, they typically treat each Gaussian as an indivisible unit, limiting their ability to exploit heterogeneous redundancy across geometry, appearance, and structure.

Inspired by compaction approaches [19, 47, 13, 2, 55, 56, 46], we aim to directly eliminate redundancy in the data and transform the original 3DGS into a pyramidal representation by introducing only one additional step during training. In the proposed representation, each layer consists of a mixture of a finite number of Gaussian components, each of which captures the statistics of a single cluster. Each cluster can be viewed as a directional and scale-aware probabilistic entity, with its center encoding a compositional representation of the attributes shared by the Gaussians within the cluster. Specifically, the pyramid starts from trained 3DGS (L_0) and is reduced progressively upward: L_1 clusters geometric attributes (position and shape), and L_2 clusters SH appearance attributes under the L_1 grouping constraints. The compact representation retains only L_1 and L_2, the resulting cluster centers can therefore be interpreted as compressed shared attribute representations for local regions. Furthermore, the ratio allocation across layers is jointly predicted from the hierarchical scale and the target compression ratio, enabling a better trade-off between representation capacity and compression efficiency while naturally supporting controllable compression.

From a modeling perspective, the optimization of clusters in each layer can be formulated as parameter estimation in a latent-variable model. By maximizing the latent-variable likelihood with EM, we obtain both the cluster-center parameters and the corresponding cluster statistics. However, the EM solution is essentially a maximum-likelihood parameterization of Gaussian attribute distributions: while it fits the data distribution well, it does not directly guarantee optimal rendering quality. To address this gap, we further derive the functional mapping relationship before and after attribute transformation under a rendering-consistency constraint, and obtain an approximate closed-form solution via moment matching to refine the opacity and geometric attributes of the cluster centers. This refinement better preserves structured scene information under the compressed representation, and is particularly beneficial for modeling edges, fine structures, and occlusion relationships.

The main contributions of this work are as follows:

- We propose Pyramidal Gaussian Mixture Splatting (PGMS), a backbone-agnostic framework built upon 3DGS that models the hierarchy of scale-dependent attributes. By progressively constructing the mixture primitive pyramid (MPP), PGMS reduces the joint redundancy of geometry and appearance while preserving their coupled distribution, thereby retaining critical structural and visual information even under high compression.
- We introduce a lightweight rate-adaptive capacity allocator for PGMS initialization. Implemented as a permutation-invariant set encoder, it predicts the budget allocation across different pyramid levels under a target reduction ratio, enabling high rendering quality under a specified parameter budget.
- We derive a compositing-aware update rule for the reduced primitives in PGMS. By reformulating the rendering objective so that the rendered pixels of PGMS approximate those of the original dense primitives, the proposed update aggregates visual attributes from dense Gaussians across all pixels, enabling efficient PGMS optimization while maintaining high rendering quality.

Experiments were performed on Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets, and results were compared to compaction and compression baselines using metrics such as PSNR, SSIM, LPIPS, and reduced file size. Compared with compaction methods, PGMS outperforms the closest competitor by up to 0.43 dB at a similar storage size under matched encoding conditions. Compared with compression methods, PGMS delivers superior rendering quality while maintaining competitive storage efficiency across different backbones.

2 Related Work

3DGS compaction. 3DGS compaction generally operates on a pre-trained representation and is largely backbone-agnostic, e.g., vanilla 3DGS [23] or Mini-Splatting [14]. Most methods reduce primitives through pruning, sparsification, or reparameterization based on per-Gaussian importance estimation. Representative examples include Taming 3DGS [34], LP-3DGS [56], Trimming the Fat [2], PUP-3DGS [19], MaskGaussian [32], GaussianSpa [55], and SteepGS [46], which mainly differ in how they derive retention or removal scores from rendering statistics, sensitivity, or density variation. In contrast, GHAP [47] formulates the whole scene as a Gaussian mixture model [41] and performs globally consistent geometric compaction via transport over means and covariances. However, its clustering process does not explicitly model appearance, making it difficult to preserve the coupled geometry–appearance structure. Hierarchical mixture models have also been used for part-controllable 3D generation via Gaussian splatting [52] and for part-aware 3D generation in hyperbolic space [53]. Our method follows the global clustering perspective of GHAP but introduces hierarchical probabilistic grouping that jointly models geometry and appearance, leading to a finer, compact representation and better rendering-consistent initialization.

3DGS compression. Existing 3DGS compression methods can be roughly divided into two groups. The first group compresses a fixed representation through quantization and entropy coding, such as FCGS [6], MoP [30], HEMGS [29], and FlexGaussian [45]. The second group further combines pruning, reparameterization, and coding, including Reduced-3DGS [38], EAGLES [16], LightGaussian [13], CompGS [31], MesonGS [51], DF-3DGS [11], ContextGS [48], HAC/HAC++ [7, 8], and PCGS [9]. Meanwhile, lightweight representations such as F-3DGS [44], C3DGS [27], IGS [49], Scaffold-GS [33], LocoGS [43], and CodeGS [28] reduce storage by redesigning the representation itself. Beyond compression, related work spans hierarchical 4D LiDAR generation [50] and language-guided visual tasks [36, 35]. Different from existing methods that mainly remove redundancy within a fixed or mildly reparameterized representation, our method introduces a representation-conversion step that reconstructs dense Gaussians into a pyramidal mixture representation, thereby reducing redundancy in the original data while preserving the joint geometry–appearance distribution and providing a structured prior for subsequent entropy coding.

3 Method

As illustrated in Figure 2, our framework consists of two stages. After the densification stage of Gaussian training, we first construct a mixture primitive pyramid via hierarchical clustering to jointly exploit the geometric and appearance attributes of the dense Gaussians, while a rate-adaptive capacity

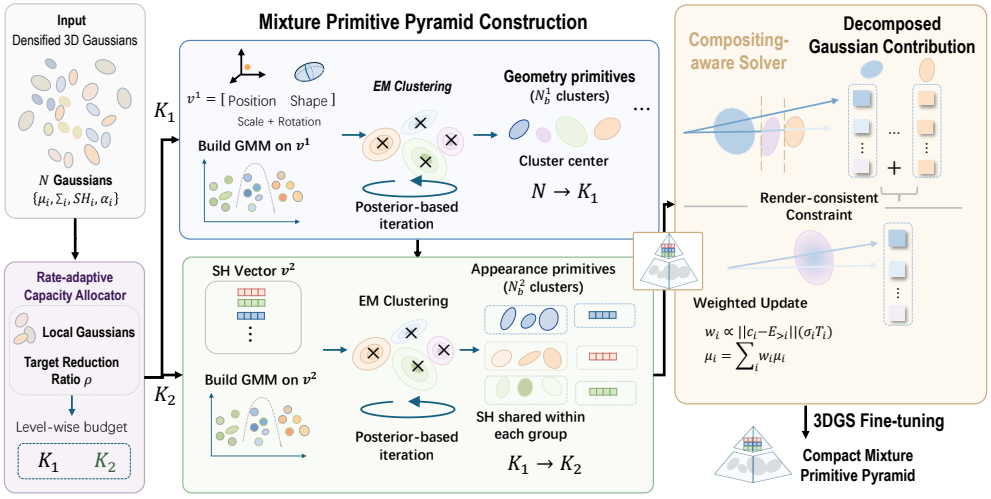


Figure 2: Starting from a trained 3DGS model, PGMS uses a pretrained allocator to divide the parameter budget between geometry and appearance. It first groups Gaussians by geometry, then clusters spherical harmonic coefficients within each group. To further preserve visual fidelity, a compositing-aware update rule is derived to refine the pyramid primitives in a single step, ensuring their rendered output closely approximates that of the original dense Gaussian set.

allocator determines the capacity of each pyramid level under a target compression ratio. We then perform a closed-form update of the pyramid attributes based on the correspondence between dense Gaussians and pyramid primitives. The resulting geometry and appearance parameters are further optimized in the remaining 3DGS fine-tuning stage.

3.1 Mixture Primitive Pyramid for Parameter Reduction

Given a trained dense Gaussian set, we first aim to reduce redundancy in the dense representation. Existing compaction methods, such as GHAP [47], reduce the number of Gaussians by aggregating primitives with similar geometric properties, yet they do not explicitly model the joint redundancy between geometry and appearance attributes. However, pixel colors are determined by the geometric and appearance attributes of both foreground and background Gaussians through alpha compositing, so even a slight geometric perturbation can alter the effective appearance contribution and thereby induce coupled redundancy in trained Gaussians. To address this issue, we introduce a mixture primitive pyramid (MPP) representation that progressively decomposes Gaussian attribute reduction into probabilistic modeling over attribute subsets. The pyramid starts from trained dense 3DGS at L_0 and progressively reduces its attributes upward into L_1 (geometry) and L_2 (SH appearance). Only these two derived layers are retained. At each level, clustering yields shared indices, and only the Gaussian subset associated with the same index is further clustered at the next level. In this way, probabilistic grouping at the previous level constrains the optimization space of the subsequent level, enabling appearance sharing and geometric refinement to be performed in a layer-wise collaborative manner.

Level-wise clustering. We first initialize the cluster centers by applying farthest-point sampling [12] to Gaussians whose opacity exceeds a predefined threshold. This strategy favors more important Gaussians while maintaining good spatial coverage. We formulate clustering using a Gaussian mixture model and solve it with the EM algorithm. In the E-step, we compute the joint log-probability of each Gaussian under each component based on the initialized cluster centers: $\ell_{ij} = \log \pi_j + \log \mathcal{N}(x_i \mid \mu_j, \mathrm{diag}(\sigma_j^2))$, and then normalize these probabilities to obtain the posterior responsibility: $r_{ij} = \exp(\ell_{ij}) / \sum_{k=1}^{M} \exp(\ell_{ik})$. In the M-step, we update the prior, mean, and diagonal variance as

$$\pi_j = \frac{N_j}{N}, \quad \mu_j = \frac{1}{N_j} \sum_{i=1}^{N} r_{ij} x_i, \quad \sigma_{jd}^2 = \frac{1}{N_j} \sum_{i=1}^{N} r_{ij} x_{id}^2 - \mu_{jd}^2, \tag{1}$$

where $N_j = \sum_{i=1}^{N} r_{ij}$. Due to the large number of Gaussians, we use KD-based partitioning [4] to build pyramids locally, thereby reducing computational overhead. After a few iterations, we obtain stable cluster centers at each level and hard-assign each Gaussian to the cluster with the highest responsibility. This hard assignment decouples dependencies across clusters, enabling subsequent clustering stages to be carried out independently across different groups.

Hierarchical attribute selection. The first-level clusters Gaussians primarily by position and shape, and the resulting cluster centers serve as the geometric primitives of the pyramid, thereby quickly removing redundant Gaussians. The L_2 clusters spherical-harmonic (SH) appearance attributes subject to the groups established by geometric clustering at L_1. During subsequent fine-tuning, the appearance primitives shared by Gaussians within the same second-level cluster are jointly optimized, enabling effective appearance compression. Overall, our method can be viewed as a heuristic hierarchical compression strategy: the first level learns parameterized geometric centers, while the second level learns a shared appearance codebook. Because sufficient Gaussian primitives are retained throughout the compression process, our MPP preserves scene structure while substantially reducing the number of Gaussian parameters.

3.2 Rate-Adaptive Capacity Allocator for MPP Instantiation

MPP defines the construction rules, the remaining step is to determine the number of components at each level, thereby instantiating the hierarchical structure. Conventional approaches enumerate candidate component counts and select the best configuration according to criteria such as BIC [42, 15] or AIC [1]. In our setting, however, the component allocations across the two levels are jointly constrained by the parameter budget and rendering quality, making exhaustive enumeration impractical. We therefore formulate this step as a hierarchical capacity-allocation problem subject to a prescribed parameter budget.

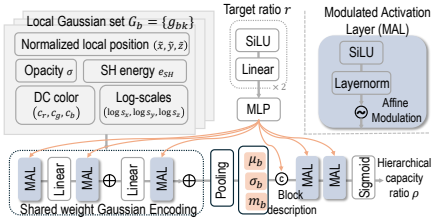


Figure 3: Rate-adaptive capacity allocator.

Specifically, we adopt a unified budgeting scheme for each block and introduce a block-level *Rate-adaptive Capacity Allocator*. For the b-th KD block, let N_b^0 denote the number of original Gaussians and ρ the user-specified retained parameter fraction. The target parameter budget for this block is defined as $B_b = 59\rho N_b^0$, where the 59 dimensions consist of 48 dimensions for third-order SH appearance parameters and 11 dimensions for Gaussian geometry, scale, rotation, and opacity parameters. Given block-level statistics, the allocator predicts a scaling factor β to allocate the budget between first-level (L_1) geometric primitive clustering and second-level (L_2) appearance-sharing clustering:

$$N_b^1 = \left\lfloor \frac{\beta B_b}{11} \right\rfloor, \qquad N_b^2 = \left\lfloor \frac{(1-\beta) B_b}{48} \right\rfloor. \tag{2}$$

We first construct N_b^1 geometric clusters at L_1 within each KD block. We then allocate the N_b^2 appearance prototypes to SH clustering subject to these L_1 grouping constraints.

Model Architecture. In Figure 3, for the b-th block, let $G_b = \{g_{bk}\}_{k=1}^{N_b^0}$ denote the input Gaussian set. Each Gaussian feature $g_{bk} \in \mathbb{R}^{11}$ is composed of normalized local coordinates $(\tilde{x}, \tilde{y}, \tilde{z})$, log-scales $(\log s_x, \log s_y, \log s_z)$, opacity o, DC color (c_r, c_g, c_b), and the high-order spherical harmonic energy $e_{\text{sh}} = \|SH_{>0}\|_2$. Since FiLM performs feature-wise affine modulation conditioned on external inputs, it is well-suited for rate-conditioned prediction. Each Gaussian is encoded by a shared three-layer modulated activation network, with FiLM applied after every layer.

Because Gaussians within the same block form an unordered set, we aggregate per-Gaussian features using mean, max, and standard-deviation pooling, and concatenate the block-size statistic $\log N_b$ to obtain the block representation $z_b = [\mu_b; m_b; \sigma_b; \log N_b]$. This permutation-invariant aggregation ensures that the block representation is insensitive to input order. Meanwhile, the target retained parameter fraction ρ is mapped by a rate encoder composed of SiLU and linear layers into a conditioning vector e_r, which is further transformed by linear layers into modulation parameters. Finally, we concatenate z_b with the conditioning vector e_r and feed them into two modulated

activation layers, producing a block-level scaling factor in $(0, 1)$. We generate five groups of FiLM [39] parameters for feature-wise affine modulation, among which three groups are used to modulate the Gaussian encoder and two groups are used to modulate the block-level prediction head. This conditional design enables a single network to adapt its feature transformation behavior across different target parameter budgets.

Shared allocator training. Our capacity allocator is a lightweight feed-forward module trained once, with no per-scene optimization at test time. We randomly collect 100k KD-tree blocks from Mip-NeRF 360. During training, we use only two retained parameter fractions, $\rho \in \{0.05, 0.3\}$. For each block and a prescribed ρ, we uniformly evaluate 20 candidate allocation ratios $\beta \in [0.1, 0.5]$ and select the one yielding the lowest global rendering error on the training views before fine-tuning as the supervision target $\beta^\star$. The image rendering loss is used only to select $\beta^\star$; we then train the allocator with a regression loss between its predicted ratio $\hat{\beta}$ and $\beta^\star$. The same shared allocator is applied at inference to previously unseen scenes and target rates without additional optimization.

3.3 Compositing-Aware Solver for Render-Consistent Attribute Update

In 3D Gaussian Splatting, the rendered color at pixel x is obtained through front-to-back alpha compositing over the projected Gaussians:

$$C(x) = \sum_{i=1}^{N} c_i \alpha_i T_i, \tag{3}$$

where $\alpha_i = \sigma_i \cdot \exp\left(-\frac{1}{2}(x-\mu_i)^T(\Sigma_i)^{-1}(x-\mu_i)\right)$ denotes the effective opacity of the i-th Gaussian at pixel x, and $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$ is the transmittance. Here, μ_i and Σ_i represent the mean and covariance of the Gaussian projected onto the 2D image plane. Note that we omit the prime notation on α, μ, and Σ to avoid subscript conflicts with subsequent equations.

By expanding the Gaussian term on the right-hand side and rewriting it as a matrix product between a pixel-dependent basis function and a Gaussian-parameter-dependent coefficient vector, we obtain:

$$\alpha = \sigma \cdot \exp(\phi(x) \cdot \Phi_i). \tag{4}$$

where $\phi(x) \in \mathbb{R}^6$ is a quadratic feature vector containing the terms $(u^2, v^2, uv, u, v, 1)$, and $\Phi_i \in \mathbb{R}^6$ is the corresponding Gaussian parameter vector composed of the quadratic coefficients derived from the inverse covariance matrix entries (a_i, b_i, c_i) and the projected center (u_i, v_i).

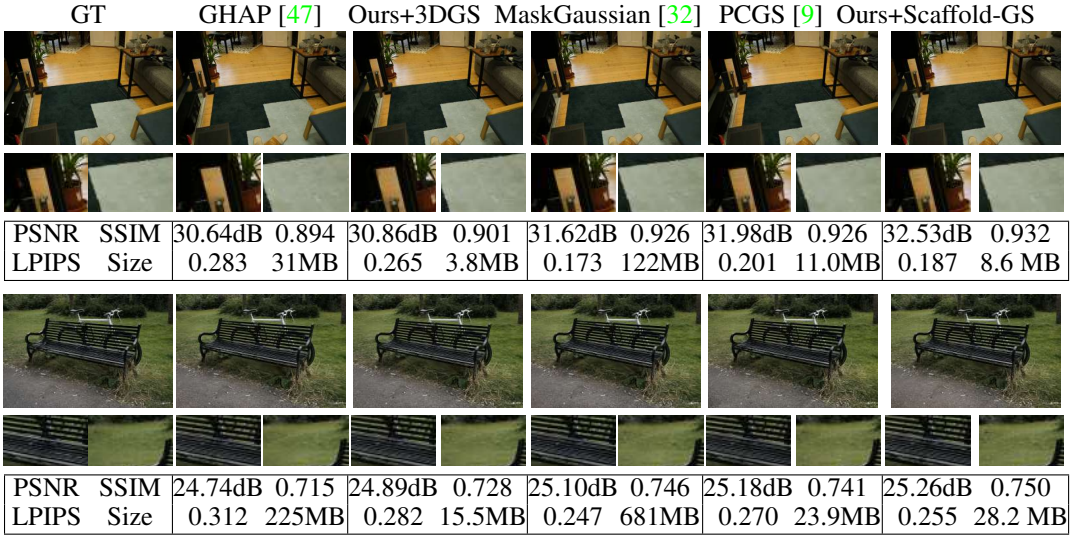
We aim to replace the original Gaussians $(1, 2, \ldots, n)$ within a local region using a single new Gaussian (indexed as 1) across the entire rendered image. During rendering, this new Gaussian should reproduce the layer-by-layer compositing effect of all original Gaussians:

$$c'_{n1}\alpha'_{n1}\prod_{j=1}^{0}(1-\alpha_j) = \sum_{i=1}^{n} c_{oi}\alpha_{oi}\prod_{j=1}^{i-1}(1-\alpha_{oj}), \tag{5}$$

We derive a closed-form approximate solution using moment matching, and aim to match the first-order moment such that variations in the local neighborhood remain consistent, i.e., we want $\nabla_\phi C_n(\phi)\big|_{\phi=0}$ and $\nabla_\phi C_o(\phi)\big|_{\phi=0}$ to be as close as possible. Since the quantity to match is a 3×6 Jacobian matrix, whereas the unknown parameter is a 6-dimensional vector, directly enforcing exact equality typically leads to an overdetermined system. We therefore adopt a least-squares formulation by minimizing $\|\nabla_\phi C_n(\phi) - \nabla_\phi C_o(\phi)\|_F^2$, which yields the following closed-form update for Φ'_{n1}:

$$\Phi'_{n1} = \frac{\sum_{k=1}^{n}\left[\sigma_{ok}T_k(c_{ok} - E_{>k})\right]\Phi_{ok}}{c'_{n1}\sigma'_{n1}}, \quad \text{where } E_{>k} = \sum_{i=k+1}^{n} c_{oi}\sigma_{oi}\prod_{j=k+1}^{i-1}(1-\sigma_{oj}). \tag{6}$$

This rule aggregates the render-consistent contributions from all original Gaussians, where each term combines the effective visibility of Gaussian $\sigma_{ok}T_k$ with its net color residual $(c_{ok} - E_{>k})$. Specifically, c_{ok} represents the color directly contributed by the k-th Gaussian as an emitter, whose weight is determined by its current visibility $\sigma_{ok}T_k$. $E_{>k}$ captures the role of the k-th Gaussian as an



PSNR SSIM	30.64dB 0.894	30.86dB 0.901	31.62dB 0.926	31.98dB 0.926	32.53dB 0.932
LPIPS Size	0.283 31MB	0.265 3.8MB	0.173 122MB	0.201 11.0MB	0.187 8.6 MB

PSNR SSIM	24.74dB 0.715	24.89dB 0.728	25.10dB 0.746	25.18dB 0.741	25.26dB 0.750
LPIPS Size	0.312 225MB	0.282 15.5MB	0.247 681MB	0.270 23.9MB	0.255 28.2 MB

Figure 4: Qualitative comparison with GHAP, MaskGaussian, and PCGS. The second and third columns compare methods built on the 3DGS backbone under high-compression settings, and the last three columns correspond to lower compression ratios, where reconstruction quality becomes the primary goal. Our PGMS is evaluated on both 3DGS and Scaffold-GS backbones.

occluder: by blocking light rays that would otherwise be transmitted, it reduces the contribution of all Gaussians behind it to the final color.

Although Φ provides a pixel-wise matching view, it is tightly coupled with the Gaussian parameters, and updating Φ first then back-solving often leads to numerical instability. We instead apply Φ-derived weights to update μ and Σ via weighted aggregation of individual covariances, preserving the alpha-compositing occlusion structure while avoiding instability from inverse matrix solving.

4 Experiments

4.1 Implementation details

Following prior work [9, 45, 47], we adopt Mip-NeRF 360 [3], Tanks & Temples [25], and Deep Blending [20] as benchmark datasets. We evaluate performance using PSNR, SSIM, LPIPS, and the final storage size of the compressed representation. We first present the backbone performance of vanilla 3DGS [23], Scaffold-GS [33], Mini-Splatting [14], and the compact representation LocoGS [43]. We then compare our method against both *compaction* and *compression* approaches.

For compaction baselines, we include Trimming the Fat [2], LightGaussian [13], 3DGS-MCMC [24], GHAP [47], PUP-3DGS [19], and MaskGaussian [32]. Since the first 5 methods support adjustable compaction ratios, we follow GHAP and compare them under a unified 90% compaction setting. Specifically, all methods start from the same vanilla 3DGS model trained for 15k iterations. Each method then applies its own compaction module to remove 90% of the Gaussians, followed by another 15k iterations of finetuning on the geometry and appearance parameters of the remaining primitives. For compression baselines, we compare with HAC++ [8], PCGS [9], OMG [26], and FlexGaussian [45]. We also include GHAP at a lower compaction ratio in this group for reference. We apply our compression method to vanilla 3DGS, Scaffold-GS, and Mini-Splatting. Leaf-level geometry is encoded with G-PCC [18, 17], while scales, rotations, opacity, and SH indices are quantized and losslessly compressed. SH prototypes are grouped by frequency band, quantized with band-specific bit widths, and similarly compressed. We exclude the total 143 KB footprint of the shared Capacity Allocator from the compressed file size, as this is a negligible average overhead of only 11 KB per scene across our three datasets.

Table 1: Comparison on Mip-NeRF 360, Tanks & Temples and Deep Blending with Size/Count Metrics.

	Method	Backbone	Mip-NeRF 360				Tanks & Temples				Deep Blending			
			SSIM↑	PSNR↑	LPIPS↓	Size	SSIM↑	PSNR↑	LPIPS↓	Size	SSIM↑	PSNR↑	LPIPS↓	Size
	3DGS [23]		0.813	27.55	0.221	746.5	0.853	23.79	0.169	433.2	0.907	29.82	0.238	699.2
	Scaffold-GS [33]		0.806	27.50	0.252	187.3	0.853	23.96	0.177	154.3	0.906	30.21	0.254	121.2
	MiniSplatting [14]		0.759	26.02	0.318	119.5	0.799	22.66	0.265	67.6	0.895	29.40	0.289	124.9
	LocoGS [43]		0.798	27.05	0.257	7.9	0.843	23.66	0.191	6.6	0.903	29.97	0.261	7.6
Flexible Ratio	PCGS* [9]	Scaffold-GS	0.810	27.81	0.235	25.6	0.850	24.26	0.181	10.3	0.905	30.07	0.262	5.9
Flexible Ratio	HAC++ [8]	Scaffold-GS	0.811	27.82	0.231	18.5	0.854	24.32	0.178	8.6	0.911	30.34	0.254	5.3
Flexible Ratio	Ours	Scaffold-GS	**0.816**	**28.00**	**0.224**	**18.2**	**0.857**	**24.39**	**0.167**	**7.6**	**0.911**	**30.50**	**0.244**	**4.7**
Flexible Ratio	FlexGaussian [45]	3DGS	0.780	26.38	0.251	40.8	0.804	22.44	0.219	16.3	0.884	28.61	0.269	25.5
Flexible Ratio	GHAP-L [47]	3DGS	0.788	26.97	0.275	149.3	0.835	23.62	0.212	86.7	0.907	29.86	0.252	139.8
Flexible Ratio	MaskGaussian [32]	3DGS	0.811	27.43	0.227	271.2	0.847	23.72	0.181	132.8	0.907	29.69	0.244	156.2
Flexible Ratio	Ours	3DGS	**0.813**	**27.52**	**0.226**	**16.3**	**0.850**	**23.77**	**0.174**	**9.8**	**0.909**	**29.92**	**0.239**	**14.5**
Flexible Ratio	OMG [26]	MiniSplatting	0.819	27.34	0.218	**6.8**	0.846	23.60	0.181	**3.9**	0.910	29.88	0.247	**5.2**
Flexible Ratio	GHAP	MiniSplatting	0.821	27.31	0.214	218.0	0.855	23.40	0.171	188.6	0.912	30.17	0.238	218.6
Flexible Ratio	Ours	MiniSplatting	**0.828**	**27.51**	**0.190**	11.7	**0.857**	**23.67**	**0.143**	9.4	**0.912**	**30.18**	**0.216**	11.9
90% Compaction	Trimming the Fat [2]	3DGS	0.731	25.26	0.343	74.7	0.776	21.54	0.293	43.3	0.887	28.06	0.302	69.9
90% Compaction	LightGaussian [13]	3DGS	0.735	25.67	0.331	74.7	0.756	22.11	0.306	43.3	0.869	28.01	0.327	69.9
90% Compaction	PUP-3DGS [19]	3DGS	0.753	25.33	0.309	74.7	0.767	21.52	0.280	43.3	0.895	29.15	0.274	69.9
90% Compaction	3DGS-MCMC [24]	3DGS	0.763	25.96	0.309	74.7	0.779	22.14	0.282	43.3	0.885	28.98	0.298	69.9
90% Compaction	GHAP-H	3DGS	0.764	26.40	0.314	74.7	0.818	23.31	0.242	43.3	0.905	29.65	0.264	69.9
90% Compaction	Ours-H	3DGS	**0.782**	**26.82**	**0.282**	**8.6**	**0.830**	**23.74**	**0.222**	**5.0**	**0.906**	**29.79**	**0.257**	**7.4**

4.2 Compression Performance

Table 1 shows that PGMS achieves a strong and consistent rate–distortion trade-off across all three benchmarks under multiple backbone choices, and in several cases even approaches or surpasses the original backbone with a smaller storage. In the Scaffold-GS and 3DGS settings, it achieves the best reconstruction quality while requiring the least storage. On the 3DGS backbone, PGMS reduces file size relative to FlexGaussian by 60% on Mip-NeRF 360, 40% on Tanks & Temples, and 43% on Deep Blending. On MiniSplatting, PGMS consistently outperforms all baselines in all quality metrics, whereas OMG remains superior in absolute storage size, likely because it is more aggressively optimized for extreme compression. Moreover, the advantage of PGMS is particularly pronounced in the 90% compaction regime, where it still delivers the best rendering quality across all datasets. Under the same encoding conditions, PGMS achieves 27.25 dB at 8.62 MB, compared with 26.82 dB at 9.43 MB for GHAP, supporting a quality advantage at a similar storage size.

Figure 4 shows a qualitative comparison with baselines. Under aggressive compression, our method achieves better rendering quality than GHAP, with clearer foreground floor boundaries and more faithful local structures, such as floor–carpet transitions and glass edges in the *room* scene and sharper bench details in the *bicycle* scene, indicating effective preservation of critical information at very low bitrates. This advantage comes from the higher parameterization efficiency of the proposed pyramidal mixture representation, which allocates the limited bit budget to the most important content. At lower compression ratios, our method recovers finer details—such as floor markings behind the glass table, subtle carpet textures, and surface scratches in the *room* scene and richer background structure in the *bicycle* scene—thanks to stronger representational capacity and a rendering-aware parameter update strategy that improves reconstruction in occluded regions and around fine structures.

Computational Efficiency. In Figure 5, we compare the computational cost of our compression pipeline against PCGS [9] on the Scaffold-GS [33] backbone. Our method combines spatial-aware GMM clustering, anchor pruning, and efficient hybrid coding to achieve significantly faster encoding and decoding with only marginal post-hoc overhead. Specifically, it provides about 5.6 $\times$ faster encoding and 21 $\times$ faster decoding, while also yielding a smaller bitstream.

4.3 Ablation Study

Component Ablation. In Table 2 we perform a progressive ablation on Mip-NeRF 360 using vanilla 3DGS as the backbone, with the parameter reduction ratio fixed at 0.2. Single-layer GMM slightly outperforms GHAP, indicating that posterior-based soft assignment yields smoother clustering and more representative centers. The pyramidal representation substantially recovers rendering quality under the same budget, showing that hierarchical modeling better preserves critical geometry and appearance. Both the Rate Allocator and Parameter Update bring consistent gains, while Entropy

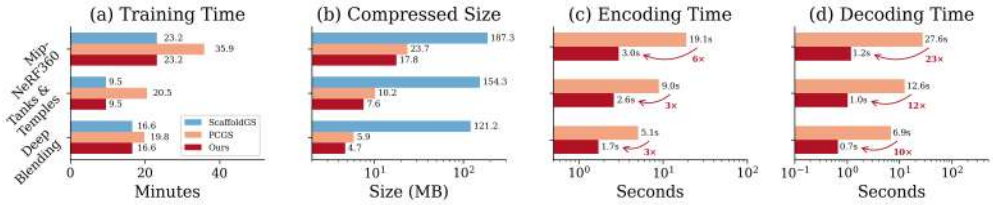


Figure 5: Computational complexity comparison with PCGS using the Scaffold-GS backbone on Mip-NeRF 360, Tanks & Temples, and Deep Blending. Our method achieves 5.6× faster encoding and 21× faster decoding than PCGS while producing 26% smaller bitstreams, with negligible additional training overhead beyond the baseline.

Table 2: Component ablations on Mip-NeRF 360 using 3DGS backbone with a fixed reduction ratio of 0.2. For the last variant, we include both the runtime and the entropy coding time.

Method	Pyramid	Allocation	Update	Entropy	PSNR↑	SSIM↑	LPIPS↓	Size↓	Time↓ (s)
3DGS [23]					27.55	0.813	0.221	746.5M	1205.0
GHAP [47]					26.97	0.788	0.275	149.3M	1209.6
GMM-GS	✓				26.99	0.790	0.276	149.3M	1208.6
Pyramidal	✓	Manual			27.28	0.807	0.238	149.3M	1208.9
+ Rate Allocator	✓	Learned			27.40	0.810	0.224	148.9M	1209.8
+ Parameter Update	✓	Learned	✓		27.53	0.815	0.216	148.9M	1209.9
+ Entropy Coding	✓	Learned	✓	✓	27.45	0.813	0.223	16.3M	1210.3

Coding further reduces storage with only minor quality loss. Training time remains largely unchanged, and the full model adds only about 5.3 s of training and coding overhead compared with the baseline—comparable to GHAP's 4.6 s—demonstrating efficient compaction with compression-level storage reduction and higher rendering quality.

Ablation on Reduction Ratio. To demonstrate the generality of our method, Table 3 compares it with the state-of-the-art compaction baseline GHAP across two representative backbones, 3DGS [23] and MiniSplatting [14], under two challenging compaction budgets, 10% and 20%. Our PGMS consistently outperforms GHAP across different backbones and reduction settings. Under a 20% budget, our method retains most of the original 3DGS information and even surpasses the backbone, including Deep Blending with the 3DGS backbone and Tanks & Temples and Deep Blending with the MiniSplatting backbone. These results suggest that our method goes beyond retaining a subset of the original Gaussians. Instead, it constructs pyramidal primitives that preserve the principal geometric and appearance cues while reducing redundancy and suppressing less informative components, potentially yielding a regularization effect.

Ablation on Binary Coding. We ablate the binary coding strategy for attributes beyond the SH codebook. G-PCC encodes all scalar attributes, achieving the smallest size (11.3 MB) but requiring 95.6 s encoding and 53.2 s decoding. The zstd-only baseline is over 170× faster, yet yields files 69% larger. Our hybrid scheme applies G-PCC only to geometry and zstd to the remaining attributes, resulting in 13.9 MB (23% larger than G-PCC) while being 17× faster for encoding and 21× faster

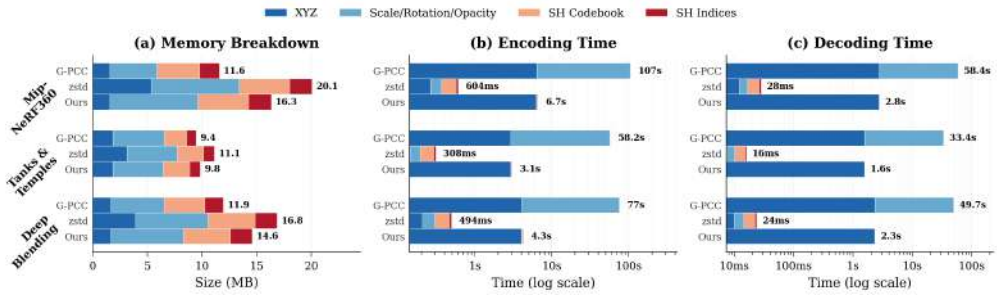


Figure 6: Compression efficiency analysis and memory breakdown of three codec configurations on Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets. For both G-PCC and our method, position occupies a disproportionately large share of the bitrate, making the bit allocation of attributes other than position barely visible.

Table 3: Comparison under different backbone ratios ρ on three datasets

Backbone	Method	ρ	Tanks & Temples			Mip-NeRF 360			Deep Blending		
			SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
3DGS	GHAP	10%	0.818	23.31	0.242	0.764	26.40	0.314	0.905	29.65	0.264
	Ours	10%	0.830	23.74	0.222	0.782	26.82	0.282	0.906	29.79	0.257
	GHAP	20%	0.835	23.62	0.212	0.788	26.97	0.275	0.907	29.86	0.252
	Ours	20%	0.846	23.74	0.189	0.813	27.25	0.253	0.908	29.88	0.244
		100%	0.853	23.79	0.169	0.813	27.55	0.221	0.907	29.82	0.238
Mini-Splatting	GHAP	10%	0.835	23.23	0.198	0.802	27.09	0.250	0.909	30.04	0.254
	Ours	10%	0.846	23.55	0.174	0.808	27.23	0.230	0.912	30.17	0.238
	GHAP	20%	0.855	23.40	0.171	0.821	27.31	0.214	0.912	30.17	0.238
	Ours	20%	0.857	23.67	0.143	0.828	27.51	0.190	0.912	30.18	0.216
		100%	0.848	23.34	0.140	0.832	27.49	0.176	0.907	29.98	0.211

for decoding. This favorable trade-off arises because the Gaussian mixture pyramid already captures local coherence and removes most geometry–appearance redundancy, making expensive spatial transforms (e.g., RAHT) less beneficial and allowing lightweight zstd to suffice for non-geometric attributes, with G-PCC reserved for spatial coordinates.

5 Conclusion

In this paper, we introduced Pyramidal Gaussian Mixture Splatting (PGMS), a backbone-agnostic framework for compact 3DGS representation. By progressively aggregating Gaussian attributes and optimizing allocation across pyramid levels, PGMS preserves critical geometric and appearance information under strong compression. Our rendering-aware parameter update further improves fidelity, especially in occluded and fine-structure regions, by aggregating information from dense Gaussians according to their image-space contributions. Experiments on both vanilla and lightweight 3DGS backbones show that PGMS achieves a favorable trade-off among rendering quality, compression ratio, and overhead, suggesting that hierarchical mixture-based compaction is a promising route toward lightweight and storage-efficient 3DGS representations.

Limitations. We evaluate PGMS on static scenes, where complex occlusions and depth discontinuities can reduce the accuracy of the compositing-aware update. The allocator selects its training targets before fine-tuning, so the preferred allocation may change as the scene parameters are refined.

6 Acknowledgement

This research was supported by the Australian Research Council (ARC) under discovery grant project # 240101926. Professor Ajmal Mian is the recipient of an ARC Future Fellowship Award (project # FT210100268) funded by the Australian Government. This work was supported in part by the National Natural Science Foundation of China under Grant (62373293, 62463020, 62472336, 62302369, U2570205); in part by the Science and Technology Project of Jiangxi province (20261BCE310009, 20261BCE310010, 20252BAC200185), in part by Young Science and Technology Star Program of Shaanxi Province (2025ZC-KJXX-25) and in part by Ganpo Talent Support Plan under Grant gpjc20260121.

References

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[2] Salman Almuhammad Alali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning. In *35th British Machine Vision Conference 2024*, 2024.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.

[4] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] Jialei Chen, Xin Zhang, Mobarak I Hoque, Francisco Vasconcelos, Danail Stoyanov, Daniel S Elson, and Baoru Huang. Surgicalgs: Dynamic 3d gaussian splatting for accurate robotic-assisted surgical scene reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 572–582. Springer, 2025.

[6] Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3d gaussian splatting compression. *arXiv preprint arXiv:2410.08017*, 2024.

[7] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pages 422–438. Springer, 2024.

[8] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac++: Towards 100x compression of 3d gaussian splatting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[9] Yihang Chen, Mengyao Li, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Pcgs: Progressive compression of 3d gaussian splatting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 3111–3119, 2026.

[10] Yann Collet. Rfc 8878: Zstandard compression and the'application/zstd'media type, 2021.

[11] Zhenqi Dai, Ting Liu, and Yanning Zhang. Efficient decoupled feature 3d gaussian splatting via hierarchical compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11156–11166, 2025.

[12] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE transactions on image processing*, 6(9):1305–1315, 1997.

[13] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

[14] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European conference on computer vision*, pages 165–181. Springer, 2024.

[15] Chris Fraley and Adrian E Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.

[16] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In *European Conference on Computer Vision*, pages 54–71. Springer, 2024.

[17] Danillo Graziosi, Ohji Nakagami, Satoru Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. An overview of ongoing point cloud compression standardization activities: Video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9:e13, 2020.

[18] MPEG PCC Group. https://github.com/mpeggroup/mpeg-pcc-tmc13, 2023. URL https://github.com/MPEGGroup/mpeg-pcc-tmc13. Accessed: 2025.

[19] Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5949–5958, 2025.

[20] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.

[21] Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11982–11992, 2025.

[22] Yiming Huang, Long Bai, Beilei Cui, Kun Yuan, Guankun Wang, Mobarak I Hoque, Nicolas Padoy, Nassir Navab, and Hongliang Ren. Surgtpgs: Semantic 3d surgical scene understanding with text promptable gaussian splatting. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 584–594. Springer, 2025.

[23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[24] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024.

[25] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[26] Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized minimal 3d gaussian splatting. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2024.

[27] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024.

[28] Soonbin Lee, Fangwen Shu, Yago Sanchez, Thomas Schierl, and Cornelius Hellge. Compression of 3d gaussian splatting with optimized feature planes and standard video codecs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25496–25505, 2025.

[29] Lei Liu, Zhenghao Chen, Wei Jiang, Wei Wang, and Dong Xu. Hemgs: A hybrid entropy model for 3d gaussian splatting data compression. *arXiv preprint arXiv:2411.18473*, 2024.

[30] Lei Liu, Zhenghao Chen, and Dong Xu. 3d gaussian splatting data compression with mixture of priors. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8341–8350, 2025.

[31] Xiangrui Liu, Xinju Wu, Pingping Zhang, Shiqi Wang, Zhu Li, and Sam Kwong. Compgs: Efficient 3d scene representation via compressed gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 2936–2944, 2024.

[32] Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 681–690, 2025.

[33] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20654–20664, 2024.

[34] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[35] Bo Miao, Mingtao Feng, Zijie Wu, Mohammed Bennamoun, Yongsheng Gao, and Ajmal Mian. Referring human pose and mask estimation in the wild. *Advances in Neural Information Processing Systems*, 37:44791–44813, 2024.

[36] Bo Miao, Weijia Liu, Jun Luo, Lachlan Shinnick, Jian Liu, Thomas Hamilton-Smith, Yuhe Yang, Zijie Wu, Vanja Videnovic, Feras Dayoub, and Anton van den Hengel. Langmap: A human-verified benchmark for hierarchical open-vocabulary goal navigation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

[37] Guy M Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company, 1966.

[38] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, 2024.

[39] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[40] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5020–5030, 2024.

[41] Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663): 3, 2009.

[42] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[43] Seungjoo Shin, Jaesik Park, and Sunghyun Cho. Locality-aware gaussian compression for fast and high-quality rendering. *arXiv preprint arXiv:2501.05757*, 2025.

[44] Xiangyu Sun, Joo Chan Lee, Daniel Rho, Jong Hwan Ko, Usman Ali, and Eunbyung Park. F-3dgs: Factorized coordinates and representations for 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 7957–7965, 2024.

[45] Boyuan Tian, Qizhe Gao, Siran Xianyu, Xiaotong Cui, and Minjia Zhang. Flexgaussian: Flexible and cost-effective training-free compression for 3d gaussian splatting. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 7287–7296, 2025.

[46] Peihao Wang, Yuehao Wang, Dilin Wang, Sreyas Mohan, Zhiwen Fan, Lemeng Wu, Ruisi Cai, Yu-Ying Yeh, Zhangyang Wang, Qiang Liu, et al. Steepest descent density control for compact 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[47] Tao Wang, Mengyu Li, Geduo Zeng, Cheng Meng, and Qiong Zhang. Gaussian herding across pens: An optimal transport perspective on global gaussian reduction for 3dgs. In *Advances in Neural Information Processing Systems*, 2025.

[48] Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex C Kot, and Bihan Wen. Contextgs: Compact 3d gaussian splatting with anchor level context model. *Advances in neural information processing systems*, 37:51532–51551, 2024.

[49] Minye Wu and Tinne Tuytelaars. Implicit gaussian splatting with efficient multi-level tri-plane representation. *arXiv preprint arXiv:2408.10041*, 2024.

[50] Zijie Wu and Na Zhao. Hierascaffold: Learning compact hierarchical representations for scalable 4d lidar generation. In *Forty-third International Conference on Machine Learning*.

[51] Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. Mesongs: Post-training compression of 3d gaussians via efficient attribute transformation. In *European conference on computer vision*, pages 434–452. Springer, 2024.

[52] Qitong Yang, Mingtao Feng, Zijie Wu, Weisheng Dong, Fangfang Wu, Yaonan Wang, and Ajmal Mian. Hierarchical gaussian mixture model splatting for efficient and part controllable 3d generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11104–11114. IEEE, 2025.

[53] Qitong Yang, Mingtao Feng, Zijie Wu, Huixin Zhu, Weisheng Dong, Yaonan Wang, and Ajmal Mian. Learning hierarchical hyperbolic mixture model for part-aware 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12695–12705, 2026.

[54] Youyi Zhan, Tianjia Shao, Yin Yang, and Kun Zhou. Real-time high-fidelity gaussian human avatars with position-based interpolation of spatially distributed mlps. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26297–26307, 2025.

[55] Yangming Zhang, Wenqi Jia, Wei Niu, and Miao Yin. Gaussianspa: An" optimizing-sparsifying" simplification framework for compact and high-quality 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26673–26682, 2025.

[56] Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. Lp-3dgs: Learning to prune 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 37:122434–122457, 2024.

[57] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21634–21643, 2024.

A Implementation details

The shared capacity allocator is trained once on 100k randomly collected KD-tree blocks from Mip-NeRF 360, using only two retained parameter fractions, $\rho \in \{0.05, 0.3\}$ and supervision selected from 20 uniformly spaced candidate allocation ratios $\beta \in [0.1, 0.5]$ by global rendering error on training views (Sec. 3.2). Its weights remain fixed when processing new scenes and target rates; allocator inference requires no per-scene optimization. Algorithm 1 illustrates the procedure for constructing the pyramid and updating its parameters. The total number of first-level geometric primitives is $\sum_{b=1}^{N_0} N_b^1$, where N_0 is the total number of KD blocks. We compress the positions of geometric primitives using G-PCC. The associated scales, rotation quaternions, opacity, and SH codebook indices are quantized and Morton-ordered [37], and then compressed using lossless coding methods. The total number of second-level SH attribute prototypes is $\sum_{b=1}^{N_0} N_b^2$. Their coefficients are grouped by frequency band, quantized to dedicated bit-widths for each band, and then compressed using lossless coding methods.

PCGS reports results trained for both 30k and 40k iterations. For fairness, we reproduce and report the 30k version using the official codebase. For the Scaffold-GS backbone, the anchors already provide a compact representation of Gaussian primitives, so we only apply single-layer clustering to the feature vectors. We compress the anchor positions using G-PCC [18, 17]. Other attributes are sorted, quantized, and compressed with zstd, and the anchor features are quantized in a manner similar to SH codebooks, while the offsets are quantized column-wise and then further compressed with zstd [10].

B Additional Visualizations

Additional visualizations are shown in Figures 1 and Figure 2. The first row compares compaction methods under a fixed high-compression setting on the 3DGS backbone, including PUP [19], GHAP [47], and our method. The second row compares methods without a fixed reduction ratio,

including MaskGaussian [32], PCGS [9], and our method, on Scaffold-GS and MiniSplatting, where reconstruction quality becomes the primary goal. Our PGMS is evaluated on 3DGS, Scaffold-GS and MiniSplatting backbones.

Algorithm 1 Mixture Primitive Pyramid

Use trained 3DGS $\mathcal{G}$ as L_0; project to v; partition into B blocks
for block $\mathcal{B}_b$ **do**
  $B_b = 59\rho N_b^0$; $\beta_b \leftarrow \mathcal{A}_\theta(\mathcal{B}_b, \rho)$ with pretrained, fixed θ
  $K_1 = \lfloor \beta_b B_b / 11 \rfloor$
  $K_2 = \lfloor (1-\beta_b) B_b / 48 \rfloor$
  L1: EM-GMM on $[\mu_i; \mathbf{s}_i] \to K_1$ geometric clusters
  for L1 cluster $j = 1 \dots K_1$ **do**
    L2: EM-GMM on SH within L1 group $j \to K_2^{(j)}$ appearance prototypes
    Update the L1 geometry using Gaussians in group j:
    $w_i^{(\phi,j)} \propto \alpha_i T_i \|c_i - E_{>i}\|_1$
    $\mu_j = \sum_i w_i^{(\phi,j)} \mu_i$
    $\Sigma_j = \sum_i w_i [\Sigma_i + (\mu_i - \mu_j)(\mu_i - \mu_j)^\top] + \epsilon \mathbf{I}$
  end for
end for

C Per-scene Results

We present the detailed breakdown of our experiments across three widely used datasets: Mip-NeRF 360, Tanks & Temples, and Deep Blending.

Table 3, Table 2, and Table 1 report the rendering quality metrics (PSNR, SSIM, and LPIPS) alongside efficiency and storage profiles (such as Size, Encode/Decode time, and Reduction time) based on the MiniSplatting, 3DGS, and Scaffold-GS backbones, respectively. These extended results demonstrate the consistent performance and generalization capability of our approach across diverse scene structures and baseline architectures.

D Additional Ablation

Choice of GMM sparsification. During pyramid construction, each Gaussian can be softly assigned to multiple clusters, introducing a trade-off among reduction speed, memory consumption, and clustering quality. Table 4 presents the ablation results of top-K sparsification in both layers of the pyramid. We use x/x to denote the top-K pair used in Layers 1 and 2, respectively. The Full/Full configuration achieves the highest post-compression quality, as retaining all cluster assignments naturally minimizes immediate information loss in the E-step. However, this trend reverses after 15K iterations of fine-tuning, where the 10/10 configuration attains the best final PSNR despite starting from a lower post-compression quality than Full/Full. This suggests that top-K sparsification acts as an implicit regularizer during compression, producing cluster centers that are more amenable to subsequent gradient-based optimization. Moreover, increasing the top-K in the first layer yields more consistent and larger gains in final PSNR than doing so in the second layer, which is consistent with the role of L_1 as the geometric clustering stage that also constrains the subsequent SH appearance clustering at L_2. Based on this trade-off, we adopt Top-10 in both layers to balance efficiency, memory usage, and final quality.

Ablation on KD-tree depth. We vary the number of spatial blocks induced by KD-tree partitioning, which controls the granularity of local GMM fitting. As shown in Figure 3, all quality metrics exhibit a concave trend. Too few blocks lead to overly complex partitions that are poorly approximated by limited-capacity GMMs, while too many blocks produce undersampled partitions that destabilize EM and degrade fidelity. A broad performance plateau spans roughly one order of magnitude in the mid-range, indicating robustness to this parameter. Clustering time increases monotonically with block count due to deeper splits and more EM instances, but remains sublinear due to batching. We select a block count within the plateau to balance quality and efficiency.

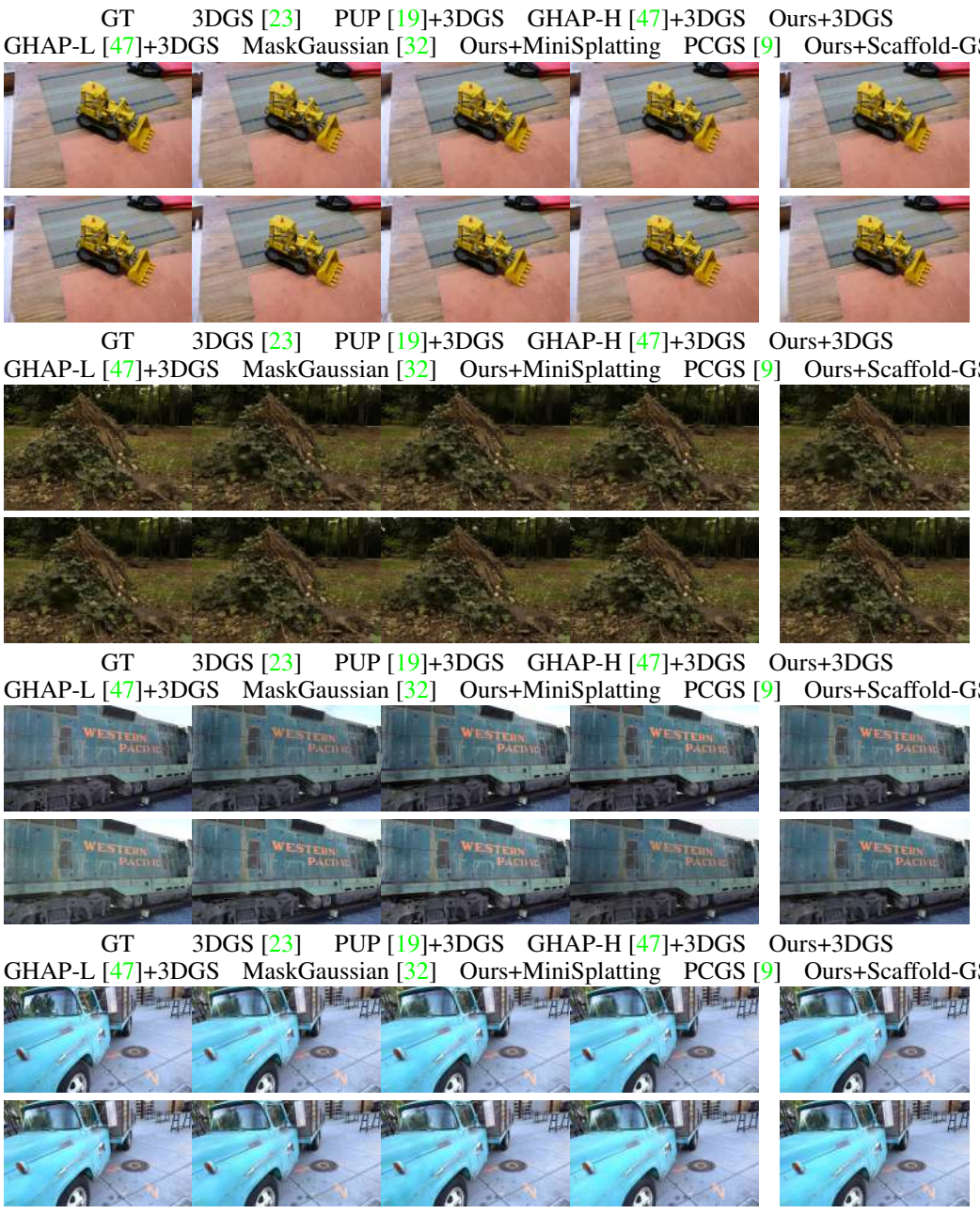


Figure 1: Qualitative comparison on Tanks & Temples and Deep Blending. The first row compares compaction methods under a fixed 90% reduction ratio on the 3DGS backbone, while the second row compares methods without a fixed reduction ratio. We evaluate our method on 3DGS, Scaffold-GS and MiniSplatting backbone, where reconstruction quality becomes the primary goal. Our PGMS is evaluated on both 3DGS and Scaffold-GS backbones.



Figure 2: Qualitative comparison on Mip-NeRF 360 dataset. The first row compares compaction methods under a fixed 90% reduction ratio on the 3DGS backbone, while the second row compares methods without a fixed reduction ratio. We evaluate our method on 3DGS, Scaffold-GS and MiniSplatting backbone, where reconstruction quality becomes the primary goal. Our PGMS is evaluated on both 3DGS and Scaffold-GS backbones.

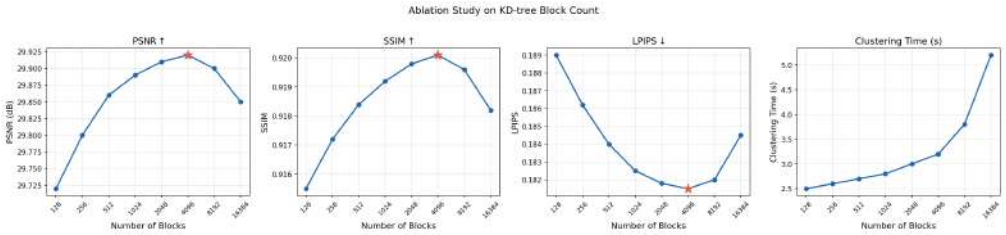


Figure 3: Ablation study on the number of KD-tree blocks for spatial partitioning. We vary the block count as powers of two and report rendering quality (PSNR, SSIM, LPIPS) and GMM clustering time on the counter scene. The red star marks the best-performing configuration.

E Derivation of First-Moment Matching for n-Gaussian Merging

E.1 Discrete Rendering Equation

Given n ordered Gaussians $\{G_{o1}, \ldots, G_{on}\}$ along a ray or a local spatial region, the rendered color at position x is computed by front-to-back alpha compositing:

$$C(x) = \sum_{i=1}^{n} c_{oi}\, \alpha_{oi}(x)\, T_{oi}(x), \qquad T_{oi}(x) = \prod_{j=1}^{i-1} \big(1 - \alpha_{oj}(x)\big). \tag{1}$$

Here, we omit the subscript on α for notational simplicity in the following derivation. Using the approximation

$$\alpha_i(x) = \sigma_i \exp\big(\phi(x)^\top \Phi_i\big), \tag{2}$$

we rewrite the rendering process as a function of the coefficient vector $\phi(x)$.

Table 1: Per-scene metrics on Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets using Scaffold-GS backbone under 50% feature reduction.

Scene	PSNR	SSIM	LPIPS	Total Size	Encode (s)	Decode (s)	Reduction (s)
Mip-NeRF 360							
bicycle	25.26	0.750	0.255	28.2	4.9	1.9	4.2
bonsai	33.32	0.951	0.176	12.6	2.1	0.8	3.7
counter	29.92	0.920	0.182	9.0	1.3	0.5	3.5
flowers	21.38	0.582	0.369	22.9	3.8	1.5	3.9
garden	27.46	0.852	0.134	23.2	3.9	1.5	4.2
kitchen	32.05	0.934	0.116	11.3	1.5	0.6	3.6
room	32.53	0.932	0.187	8.6	1.4	0.5	3.6
stump	26.81	0.770	0.257	24.6	4.4	1.7	4.0
treehill	23.30	0.649	0.341	23.4	3.9	1.5	4.2
Mean	28.00	0.816	0.224	18.2	3.0	1.2	3.9
Tanks & Temples							
train	22.81	0.830	0.195	7.4	1.8	0.7	3.6
truck	25.98	0.884	0.138	7.9	3.4	1.3	4.1
Mean	24.39	0.857	0.167	7.6	2.6	1.0	3.9
Deep Blending							
drjohnson	29.96	0.911	0.242	5.4	1.9	0.7	3.8
playroom	31.03	0.911	0.246	4.1	1.6	0.6	3.6
Mean	30.50	0.911	0.244	4.7	1.8	0.7	3.7

E.2 Parameterization of the Gaussian Term

We use the 2D quadratic basis

$$\phi(x) = \begin{bmatrix} u^2 & v^2 & uv & u & v & 1 \end{bmatrix}^\top \in \mathbb{R}^6, \tag{3}$$

with the corresponding Gaussian-dependent parameter vector

$$\Phi_i = \begin{bmatrix} a_i & c_i & 2b_i & -2a_iu_i - 2b_iv_i & -2c_iv_i - 2b_iu_i & a_iu_i^2 + 2b_iu_iv_i + c_iv_i^2 \end{bmatrix}^\top \in \mathbb{R}^6. \tag{4}$$

Here, (u_i, v_i) denotes the projected center of the i-th Gaussian on the image plane, and its inverse covariance matrix is

$$M_i = (\Sigma_i')^{-1} = \begin{bmatrix} a_i & b_i \\ b_i & c_i \end{bmatrix}. \tag{5}$$

Under the small-deviation assumption, $\alpha_i(x)$ is linearized around the Gaussian center, which yields the above parameterization.

E.3 Moment-Matching Objective

After clustering, we replace the original ordered Gaussians $\{G_{o1}, \ldots, G_{on}\}$ with a single merged Gaussian G'_{n1}. The merged primitive should reproduce the same compositing effect:

$$c'_{n1}\alpha'_{n1} = \sum_{i=1}^{n} c_{oi}\alpha_{oi}T_{oi}. \tag{6}$$

To further preserve local variation, we match the first moment in the parameter space. Specifically, we seek $(c'_{n1}, \sigma'_{n1}, \Phi'_{n1})$ such that the gradient of the merged Gaussian at $\phi = 0$ matches that of the original composite:

$$(c'_{n1}\sigma'_{n1})\,\Phi'_{n1} = \nabla_\phi C(\phi)\big|_{\phi=0}. \tag{7}$$

Table 2: Per-scene metrics on Mip-NeRF 360, Tanks & Temples, and Deep Blending using 3DGS backbone.

Scene	PSNR	SSIM	LPIPS	Hybrid Size
Mip-NeRF 360				
bicycle	25.14	0.747	0.251	28.7
bonsai	32.47	0.948	0.182	6.9
counter	29.15	0.915	0.186	6.7
flowers	21.26	0.583	0.368	16.9
garden	27.22	0.853	0.133	24.3
kitchen	31.50	0.933	0.117	9.7
room	31.68	0.928	0.198	7.7
stump	26.66	0.770	0.248	26.4
treehill	22.62	0.637	0.353	19.4
Mean	27.52	0.813	0.226	16.3
Tanks & Temples				
train	22.13	0.818	0.202	6.5
truck	25.41	0.882	0.146	13.2
Mean	23.77	0.850	0.174	9.8
Deep Blending				
drjohnson	29.48	0.907	0.236	17.8
playroom	30.37	0.911	0.241	11.2
Mean	29.92	0.909	0.239	14.5

Table 3: Per-scene metrics on Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets using MiniSplatting backbone.

Scene	PSNR	SSIM	LPIPS	Hybrid Size
Mip-NeRF 360				
bicycle	25.53	0.781	0.199	18.8
bonsai	31.27	0.948	0.160	6.8
counter	28.92	0.918	0.161	6.4
flowers	21.39	0.618	0.298	11.5
garden	27.37	0.859	0.117	14.9
kitchen	31.84	0.936	0.108	9.3
room	31.59	0.934	0.169	7.3
stump	27.27	0.804	0.203	16.5
treehill	22.47	0.652	0.293	14.0
Mean	27.51	0.828	0.190	11.7
Tanks & Temples				
train	21.66	0.827	0.182	6.2
truck	25.67	0.894	0.104	12.7
Mean	23.67	0.860	0.143	9.4
Deep Blending				
drjohnson	29.65	0.911	0.219	13.3
playroom	30.71	0.914	0.213	10.6
Mean	30.18	0.912	0.216	11.9

E.4 Gradient Decomposition

To derive the closed form, we decompose the contribution of each original Gaussian parameter Φ_k into two parts.

Emitter term. For the k-th Gaussian itself, the corresponding rendering term is

$$c_{ok}\alpha_{ok}T_{ok}. \tag{8}$$

Since T_{ok} depends only on indices $j < k$, it is independent of Φ_k. Therefore,

$$\left.\frac{\partial\big(c_{ok}\alpha_{ok}T_{ok}\big)}{\partial\phi}\right|_{\phi=0} = \left(c_{ok}\sigma_{ok}T_{ok}\right)\Phi_{ok}. \tag{9}$$

Table 4: Ablation study of top-K sparsification in PGMS on the *Garden* scene under the 3DGS backbone with a fixed reduction ratio. 'Full' denotes no sparsification (dense E-step).

L1 Top-K	L2 Top-K	Post-Compression PSNR↑	SSIM↑	LPIPS↓	After Fine-tuning PSNR↑	SSIM↑	LPIPS↓	Resources Time (s)↓	GPU (GB)↓	CPU (GB)↓
3	3	19.09	0.514	0.439	26.94	0.835	0.167	6.77	11.50	9.37
	10	19.35	0.522	0.432	26.96	0.835	0.167	7.31	11.47	9.84
	Full	19.33	0.523	0.430	26.93	0.835	0.167	6.95	11.48	9.80
10	3	19.28	0.524	0.432	27.00	0.837	0.164	**6.10**	11.49	9.83
	10	19.48	0.532	0.423	**27.01**	**0.837**	**0.164**	6.47	11.50	9.54
	Full	19.46	0.530	0.425	26.96	0.836	0.165	6.62	11.49	**9.35**
Full	3	19.24	0.524	0.431	26.97	0.837	0.165	6.34	11.50	9.93
	10	19.48	0.532	0.424	26.99	0.837	0.165	6.66	11.51	9.58
	Full	**19.54**	**0.534**	**0.422**	26.98	0.837	0.164	6.67	**11.47**	9.56

Occluder term. For any $i > k$, the parameter Φ_k also affects the transmittance through the factor $(1 - \alpha_{ok})$. Applying the product rule, we obtain

$$\left.\frac{\partial C_i}{\partial \phi}\right|_{\text{from } \Phi_k} = c_{oi}\alpha_{oi}\left(\prod_{\substack{j=1 \\ j\neq k}}^{i-1}(1-\alpha_{oj})\right)\left(-\frac{\partial \alpha_{ok}}{\partial \phi}\right). \tag{10}$$

At $\phi = 0$, where $\alpha_{oj} = \sigma_{oj}$ and $\frac{\partial \alpha_{ok}}{\partial \phi} = \sigma_{ok}\Phi_{ok}$, this becomes

$$-\, c_{oi}\sigma_{oi}\left(\frac{T_{oi}}{1-\sigma_{ok}}\right)\sigma_{ok}\Phi_{ok}. \tag{11}$$

E.5 Closed-Form Solution

Combining all terms containing Φ_k, we obtain the net contribution

$$C_k = \sigma_{ok}T_{ok}\left(c_{ok} - \sum_{i=k+1}^{n} c_{oi}\sigma_{oi}\prod_{j=k+1}^{i-1}(1-\sigma_{oj})\right). \tag{12}$$

Let

$$E_{>k} = \sum_{i=k+1}^{n} c_{oi}\sigma_{oi}\prod_{j=k+1}^{i-1}(1-\sigma_{oj}) \tag{13}$$

denote the cumulative contribution of all Gaussians after index k. Then the merged parameter admits the closed-form update

$$\Phi'_{n1} = \frac{\sum_{k=1}^{n}\left[\sigma_{ok}T_{ok}(c_{ok} - E_{>k})\right]\Phi_{ok}}{c'_{n1}\sigma'_{n1}}, \tag{14}$$

with the zeroth-moment constraint

$$c'_{n1}\sigma'_{n1} = \sum_{i=1}^{n} c_{oi}\sigma_{oi}T_{oi}, \tag{15}$$

which preserves the total color energy.

E.6 Least-Squares Interpretation

The first-moment constraint can also be viewed as a least-squares fitting problem on the Jacobian. Since the color gradient is a 3×6 matrix while Φ'_{n1} is a 6D vector, exact equality is generally overdetermined. We therefore solve

$$\min_{\Phi'_{n1}} \left\| (c'_{n1}\sigma'_{n1})(\Phi'_{n1})^{\top} - J_{\text{target}} \right\|_F^2, \tag{16}$$

where

$$J_{\text{target}} = \sum_{k=1}^{n} \Phi_{ok} \left[\sigma_{ok} T_{ok} (c_{ok} - E_{>k})\right]^{\top}. \tag{17}$$

This formulation yields the same closed-form update as above and provides a stable interpretation of the moment-matching rule.

E.7 Implementation Note

Although Φ provides a pixel-wise matching perspective, its components are strongly coupled with the Gaussian parameters. Updating Φ first and then back-solving for the parameters can introduce numerical instability. We therefore perform weighted updates directly on μ and Σ using Φ-derived weights: positions are updated by weighted averaging, and covariances are updated by aggregating individual covariances together with dispersion terms,

$$\Sigma_l' = \sum_i w_i \left[\Sigma_i + (\mu_i - \mu_l')(\mu_i - \mu_l')^{\top}\right]. \tag{18}$$

This preserves the occlusion relationships and contribution weighting implied by alpha compositing while avoiding inverse-matrix solving and back-projection. Opacity is updated using GMM responsibility weighting, which intuitively preserves cumulative transmittance.

E.8 Local Conditions and the Role of the PL Inequality

The compositing-aware update provides a local initialization for subsequent multi-view fine-tuning. Its first-order approximation assumes a fixed view and a neighborhood in which the participating Gaussians retain their depth ordering and visibility relationships. Within such a neighborhood, the alpha-compositing expression can be treated as a differentiable function of the Gaussian parameters. Depth-order changes, occlusion discontinuities, and conflicting visibility across views can reduce the accuracy of this approximation.

The Polyak–Łojasiewicz (PL) inequality gives a conditional interpretation of the refinement stage. Let $\theta^{(0)} = \theta_{\text{merge}}$ denote the compressed initialization, and let $\mathcal{L}(\theta)$ be the multi-view fine-tuning objective for the fixed compressed representation. Suppose that there is a neighborhood $\mathcal{U}$ containing the initialization in which $\mathcal{L}$ has an L_{sm}-Lipschitz continuous gradient and satisfies the local PL inequality

$$\frac{1}{2}\|\nabla \mathcal{L}(\theta)\|_2^2 \geq \mu_{\text{PL}}\big(\mathcal{L}(\theta) - \mathcal{L}^{\star}\big), \qquad \theta \in \mathcal{U}, \tag{19}$$

where $\mu_{\text{PL}} > 0$ and $\mathcal{L}^{\star}$ is the minimum objective value in the neighborhood for that representation. Here, μ_{PL} is a convergence constant, distinct from a Gaussian position.

Consider gradient descent, $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \mathcal{L}(\theta^{(t)})$, with $0 < \eta \leq 1/L_{\text{sm}}$ and $0 < \eta\mu_{\text{PL}} < 1$. Assume that all iterates and their connecting update segments remain in $\mathcal{U}$. Smoothness then gives

$$\begin{aligned} \mathcal{L}(\theta^{(t+1)}) &\leq \mathcal{L}(\theta^{(t)}) - \eta\left(1 - \frac{\eta L_{\text{sm}}}{2}\right)\|\nabla \mathcal{L}(\theta^{(t)})\|_2^2 \\ &\leq \mathcal{L}(\theta^{(t)}) - \frac{\eta}{2}\|\nabla \mathcal{L}(\theta^{(t)})\|_2^2. \end{aligned} \tag{20}$$

Combining this descent bound with Eq. 19 and applying it recursively yields

$$\mathcal{L}(\theta^{(T)}) - \mathcal{L}^{\star} \leq (1 - \eta\mu_{\text{PL}})^T \left[\mathcal{L}(\theta_{\text{merge}}) - \mathcal{L}^{\star}\right]. \tag{21}$$

Thus, under these assumptions, fine-tuning geometrically reduces the objective gap of the compressed initialization. This is a local, conditional result for gradient descent, not a global convergence guarantee for the practical 3DGS training procedure. We do not establish that the PL condition holds throughout the rendering landscape or that practical optimization remains in such a neighborhood. Moreover, $\mathcal{L}^{\star}$ need not be zero or equal to the optimum of the original dense representation: reducing the optimization gap does not guarantee recovery of all information lost through compression. The result explains why a locally recoverable initialization can benefit from further optimization, while leaving failures near occlusion boundaries and depth discontinuities possible.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 claims made.

   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: We discuss the limitations.

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: [Yes]

   Justification: The proofs appear in the supplemental material.

   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: We provide the details in the main paper and supplement.

   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: Code will be released upon acceptance; all datasets used are either publicly available or accessible upon request.

   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: We provide the details in the main paper and supplement.

   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: [Yes]

   Justification: We provide the statistical significance in supplement.

   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: We provide the details in the supplement.

   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: We confirm that the paper adheres to 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: [Yes]

    Justification: We provide the Broader impacts analysis in the supplement.

    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 poses no 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: All assets are properly credited.

    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: We will release after acceptance.

    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: [Yes]

    Justification: The paper describes the human evaluation procedure and provides general instructions. Participants were compensated following standard practices.

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: [Yes]

    Justification: The paper involves human annotators for evaluation purposes. We followed standard ethical guidelines for human evaluation.

    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: [Yes]

    Justification: The LLM is used only for editing.

    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.