Optimized Minimal 3D Gaussian Splatting

Joo Chan Lee
Sungkyunkwan University
Suwon, South Korea
maincold2@skku.edu

Jong Hwan Ko*
Sungkyunkwan University
Suwon, South Korea
jhko@skku.edu

Eunbyung Park*
Yonsei University
Seoul, South Korea
epark@yonsei.ac.kr

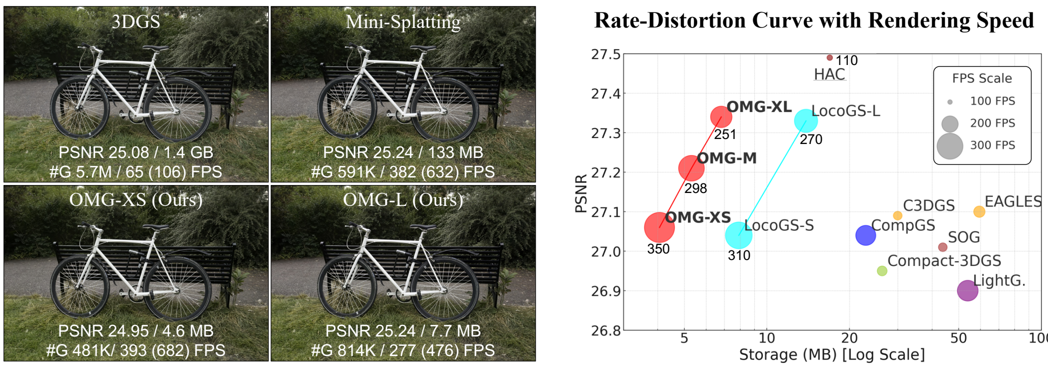

Figure 1: Our approach focuses on minimizing storage requirements while using only a minimal number of Gaussian primitives. To achieve this, we introduce a local distinctiveness metric to identify the important Gaussians. Additionally, we propose a more efficient attribute representation, particularly for sparse Gaussians, by exploiting their irregularity and continuity. As a result, our method enables scene representation under 5 MB while achieving 600+ FPS rendering.

Abstract

3D Gaussian Splatting (3DGS) has emerged as a powerful representation for real-time, high-performance rendering, enabling a wide range of applications. However, representing 3D scenes with numerous explicit Gaussian primitives imposes significant storage and memory overhead. Recent studies have shown that high-quality rendering can be achieved with a substantially reduced number of Gaussians when represented with high-precision attributes. Nevertheless, existing 3DGS compression methods still rely on a relatively large number of Gaussians, focusing primarily on attribute compression. This is because a smaller set of Gaussians becomes increasingly sensitive to lossy attribute compression, leading to severe quality degradation. Since the number of Gaussians is directly tied to computational costs, it is essential to reduce the number of Gaussians effectively rather than only optimizing storage. In this paper, we propose Optimized Minimal Gaussians representation (OMG), which significantly reduces storage while using a minimal number of primitives. First, we determine the distinct Gaussian from the near ones, minimizing redundancy without sacrificing quality. Second, we propose a compact and precise attribute representation that efficiently captures both continuity and irregularity among primitives. Additionally, we propose a sub-vector quantization technique for improved irregularity representation, maintaining fast training with a negligible codebook size. Extensive experiments demonstrate that OMG reduces storage requirements by nearly 50% compared to the previous

*Corresponding authors

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

state-of-the-art and enables 600+ FPS rendering while maintaining high rendering quality. Our source code is available at https://maincold2.github.io/omg/.

1 Introduction

3D Gaussian Splatting (3DGS) [28] has gained popularity for fast and photorealistic 3D scene reconstruction and rendering, offering a compelling alternative to conventional methods. By leveraging tile-based parallelism to approximate NeRF's [40] volumetric rendering, 3DGS enables significantly accelerated rendering while maintaining high visual quality. This has facilitated a wide range of applications, such as dynamic scene reconstruction [61, 58], photorealistic avatar generation [48, 41], generative models [56, 11], and city-scale rendering [29, 49], demonstrating its versatility across various domains.

3DGS adjusts the number of Gaussian primitives during training by iteratively cloning or splitting Gaussians with high positional gradients while removing low-opacity Gaussians. However, this optimization process introduces a substantial number of redundant Gaussians (over 3 million per 360 scenes [2]), leading to excessive storage requirements and computational overhead. To address this issue, various approaches have been proposed, including pruning based on rendering loss [32, 36] or importance score [46, 14] and optimized densification strategies [39]. Notably, several methods [14, 15, 64] reduce the number of Gaussians to around 0.5 million, enabling real-time rendering even on low-capacity GPUs while preserving rendering quality.

Despite these efforts, reducing the number of Gaussians alone does not sufficiently mitigate storage overhead. Each Gaussian is parameterized by 59 learnable parameters, so even with a reduced number of primitives, storage consumption remains substantial (e.g., 133 MB in Figure 1). To address this, many works have explored compressing Gaussian attributes by leveraging vector quantization [44, 13], neural fields [53], sorting mechanisms [42], and entropy optimization [8, 57], demonstrating considerable improvements in reducing storage consumption.

However, the aforementioned compression methods typically rely on a large number of Gaussians (over one million). This is due to two major challenges when the number of Gaussians is drastically reduced: 1) each Gaussian needs to represent a larger portion of the scene, making it more susceptible to compression loss, and 2) the increased spacing between Gaussians disrupts spatial locality, leading to higher attribute irregularity and posing challenges for entropy minimization and efficient compression. Since the number of Gaussians directly impacts computational costs, including training time and rendering speed, it is crucial to develop approaches that effectively minimize the number of Gaussians while maintaining compressibility.

In this paper, we propose Optimized Minimal Gaussian representation (OMG), an efficient compression framework that operates with a minimal number of primitives. To address the irregularity of sparse Gaussians and maximize the compressibility, we employ per-Gaussian features in a novel way. Although the reduced number of Gaussians leads to a decrease in local continuity, we can still leverage the spatial correlation associated with each Gaussian's position. Therefore, we introduce a lightweight neural field model with negligible parameters to capture the coarse spatial feature. This feature is integrated with the per-Gaussian features to represent each attribute, as shown in Figure 2. This approach requires fewer per-Gaussian parameters than directly learning the original attributes, enabling a more compact representation. While the proposed OMG architecture effectively represents sparse Gaussians, the use of per-Gaussian features impacts storage efficiency. To mitigate this, we introduce a sub-vector quantization (SVQ, Figure 3(c)), which splits the input vector into multiple sub-vectors and applies vector quantization to each sub-vector. This approach alleviates the computational overhead associated with large vector quantization codebooks (Figure 3(a)) and reduces the storage burden caused by the multiple indexing stages of residual vector quantization (Figure 3(b)), while maintaining high-precision representation.

Finally, to retain only the minimal number of Gaussians, we introduce a novel importance metric that evaluates each Gaussian's local distinctiveness relative to its neighbors, identifying the most informative Gaussians. This metric is used alongside existing importance scoring methods based on blending weights from training views [46, 14], further reducing the number of Gaussians while preserving scene fidelity.

Extensive experimental results demonstrate that OMG achieves a 49% reduction in storage compared to the previous state-of-the-art method [53], requiring only 4.1MB for the Mip-NeRF 360 dataset [2] while preserving comparable rendering quality. Additionally, OMG utilizes only 0.4 million Gaussians, enabling 600+ FPS rendering. These results underscore the effectiveness of OMG in both compression efficiency and computational performance, demonstrating it as a highly promising approach for 3D Gaussian Splatting representation.

2 Related work

2.1 Neural Radiance Fields

Neural Radiance Fields (NeRF) [40] introduced a pioneering approach for novel view synthesis by leveraging volumetric rendering in conjunction with multi-layer perceptrons (MLPs) to model continuous 3D scenes. While NeRF achieves high-quality rendering, its reliance on MLP leads to inefficiencies, particularly in terms of slow training and inference times. To overcome these limitations, the following methods [16, 34] utilized explicit voxel-based representations, enabling significantly faster training compared to traditional MLP-based NeRF models. However, these approaches still suffer from slow inference speeds and impose substantial memory requirements, posing challenges for scalability and practical deployment in large-scale environments.

Compact representation. To mitigate the memory overhead while maintaining rendering fidelity, various works have been introduced, including grid factorization [6, 5, 17, 22, 18], hash grids [43, 7], grid quantization [55, 52], and pruning-based strategies [50]. Nevertheless, achieving real-time rendering for complex, large-scale scenes remains a formidable challenge. The fundamental limitation of these approaches stems from the necessity of dense volumetric sampling, which, despite optimizations, continues to constrain training and inference speed.

2.2 3D Gaussian Splatting

Recently, 3D Gaussian Splatting (3DGS) [28] has emerged as a paradigm-shifting technique for real-time neural rendering by representing a scene with 3D Gaussian primitives. 3DGS leverages customized CUDA kernels and optimized algorithms to achieve unparalleled rendering speed while preserving high image quality. Unlike volumetric methods that require dense per-ray sampling, 3DGS projects Gaussians onto the image plane and rasterizes them tile-wise, significantly improving computational efficiency. Due to its versatility, 3DGS has become a dominant paradigm in 3D representation, leading to advancements across various domains and applications, such as mesh extraction [21, 24], simultaneous localization and mapping (SLAM) [27], dynamic scene representation [38], multi-resolution rendering [62], and further improvements in rendering quality [14]. However, 3DGS requires a substantial number of Gaussians to maintain high-quality rendering. Furthermore, each primitive is represented with multiple attributes, such as covariance matrices and spherical harmonics (SH) coefficients, requiring a large number of learnable parameters. Consequently, 3DGS demands substantial memory and storage resources, often exceeding 1GB per scene in high-resolution environments.

Reducing the number of primitives. To alleviate the substantial computational and memory overhead of 3DGS, numerous methods have been proposed to reduce the number of Gaussians while preserving rendering quality. Several approaches follow 3DGS by pruning low-opacity Gaussians, incorporated with opacity regularization [44], anchored Gaussians [37], or hyperparameter search [42]. An alternative approach utilized binary masking techniques [32, 8, 54, 57, 59], where pruning decisions are directly learned based on rendering loss. To optimize the binary masks, Compact-3DGS [44] initially adopted STE [4], while subsequent works [65, 36] employed Gumbel-Softmax.

Another direction focuses on importance-based metrics to identify and remove redundant Gaussians. These methods primarily leverage each Gaussian's blending weight contribution to rendering training-view images as a measure of importance [13, 14, 46, 19, 39]. LightGaussian [13] further incorporates Gaussian volume and opacity into the importance computation, while Taming 3DGS [39] integrates multiple information, including gradients, pixel saliency, and Gaussian attributes. Building upon these advancements, we introduce a novel importance metric that incorporates color distinction among neighboring Gaussians, enabling more effective selection of essential primitives.

Attribute compression. Earlier methods employed conventional compression techniques such as scalar and vector quantization (VQ) [13, 44, 45, 42, 19, 47, 60] and entropy coding [45, 42, 32, 8, 9] to reduce storage requirements. VQ-based representations have proven highly efficient by the fact that many Gaussian attributes are redundant across a scene, allowing for compact encoding. However, a large codebook leads to substantial computational overhead, increasing training time. While residual vector quantization (R-VQ) [32] can alleviate computational costs, it introduces additional storage inefficiencies due to the need for multiple code indices.

Another line of work explored structured representations, incorporating anchor-based encoding [37, 8, 59, 35] and factorization techniques [54], integrated with grid representations. Scaffold-GS [37] first introduced an anchor-based approach, where attributes of grouped Gaussians are encoded using shared anchor features and MLP-based refinements. Building upon this, subsequent methods [8, 10, 59, 35] incorporated context modeling to further improve compression rates. While showing high compression performance, these approaches require per-view processing, involving multiple MLP forward passes, which results in significant rendering latency.

Recent efforts have utilized neural field architectures to exploit the local continuity of neighboring Gaussians. Compact-3DGS [32] encodes view-dependent color, while LocoGS [53] represents all Gaussian attributes except for view-independent color. However, unlike NeRF-based representations, where exact spatial positions are used as inputs to neural fields, mapping Gaussian center points to their corresponding attributes remains challenging. This difficulty leads to the use of large neural field models to achieve accurate reconstruction. In this work, we propose a novel approach that effectively captures both the continuity and irregularities across Gaussians, enabling a more efficient and compact attribute representation.

3 Method

Background. 3DGS represents a scene using a set of N Gaussians, parameterized by their attributes: center position $p \in \mathbb{R}^{N\times 3}$, opacity $o \in [0,1]^N$, 3D scale $s \in \mathbb{R}_+^{N\times 3}$, 3D rotation represented as a quaternion $r \in \mathbb{R}^{N\times 4}$, and view-dependent color modeled using spherical harmonics (SH) coefficients $h^{(0)} \in \mathbb{R}^{N\times 3}$ (0 degree for static color), $h^{(1,2,3)} \in \mathbb{R}^{N\times 45}$ (1 to 3 degrees for view-dependent color). The covariance matrix of each Gaussian $\Sigma_n \in \mathbb{R}^{3\times 3}$ is determined by scale s_n and rotation r_n attributes.

To render an image, 3D Gaussians are projected into 2D space. Each pixel color in the image $C(\cdot)$ is then rendered through the alpha composition using colors c_n (determined by spherical harmonics under the given view direction) and the final opacity in 2D space $\alpha_n(\cdot)$,

$$C(x) = \sum_{k=1}^{\mathcal{N}(x)} c_k \alpha_k(x) \prod_{j=1}^{k-1} (1 - \alpha_j(x)), \tag{1}$$

$$\alpha_n(x) = o_n \exp\left(-\frac{1}{2}(x - p'_n)^T {\Sigma'}_n^{-1} (x - p'_n)\right), \tag{2}$$

where x denotes a pixel coordinate and Σ'_n, p'_n are the projected Gaussian covariance and center position. $\mathcal{N}(x)$ represents the number of Gaussians around x, where the Gaussians are depth-sorted based on the given viewing direction.

3.1 Overall architecture

OMG is designed to accurately and efficiently represent the attributes of the minimal Gaussian primitives. Existing approaches [32, 53] have leveraged neural fields to exploit the local continuity of Gaussian attributes. While effective in dense representations, this assumption weakens as Gaussians become sparser. In sparse settings, neighboring Gaussians are further apart, and smooth transitions between them become insufficient to maintain fidelity. Especially for geometry, each Gaussian covers a larger spatial region, requiring a more specific scale and rotation to accurately capture structural details. Therefore, we retain the per-Gaussian parameterization for scale $s \in \mathbb{R}_+^{N\times 3}$ and rotation $r \in \mathbb{R}^{N\times 4}$ as in 3DGS.

For appearance, local continuity can still be maintained even with increased sparsity. However, unlike NeRF, where the query input is a direct spatial point, mapping Gaussian center points to

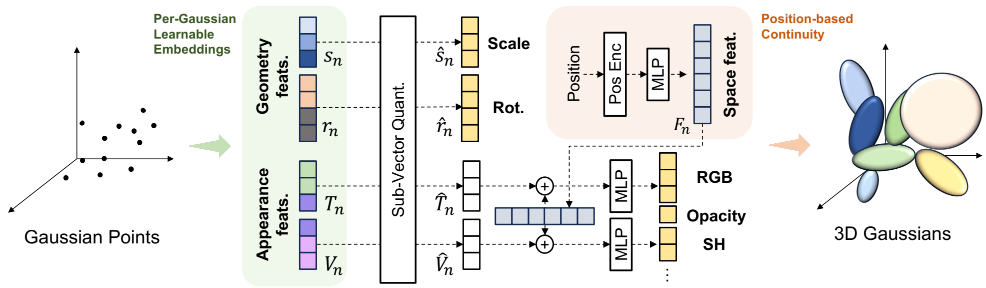

Figure 2: The overall architecture of our proposed OMG. OMG learns per-Gaussian geometric and appearance features, applying Sub-Vector Quantization (SVQ) to all of them. The SVQ-applied geometric attributes are used for rendering, while the space feature based on the Gaussian center position is integrated into the appearance features to define the final appearance.

corresponding appearances is inherently challenging. This requires a larger neural field model to maintain high fidelity. On the other hand, entirely disregarding local continuity leads to an inefficient representation, limiting the ability to capture meaningful spatial relationships. OMG addresses these challenges by integrating per-Gaussian attributes with a small neural field structure, effectively leveraging both irregularity and continuity.

To represent appearance, we learn per-Gaussian attributes, namely a static feature $T \in \mathbb{R}^{N\times D}$ and a view-dependent feature $V \in \mathbb{R}^{N\times D}$, where D is the dimensionality of each feature. As illustrated in Figure 2, each feature is concatenated with the space feature F_n, derived from each Gaussian's center position, to generate static and view-dependent color, and opacity. The space feature itself is efficiently parameterized using positional encoding and an MLP, ensuring a highly compact representation. Formally, this process can be expressed as follows:

$$h_n^{(0)} = \text{MLP}_t\big(\text{cat}(T_n, F_n)\big),\ o_n = \text{MLP}_o\big(\text{cat}(T_n, F_n)\big), \tag{3}$$

$$h_n^{(1,2,3)} = \text{MLP}_v\big(\text{cat}(V_n, F_n)\big),\ F_n = \text{MLP}_s\big(\gamma(p_n)\big), \tag{4}$$

where $\text{cat}(\cdot,\cdot)$ denotes the concatenation function, $\gamma(\cdot)$ represents the positional encoding function, and $\text{MLP}_t(\cdot), \text{MLP}_o(\cdot), \text{MLP}_v(\cdot), \text{MLP}_s(\cdot)$ are the MLPs for static color, opacity, view-dependent color, and space feature, respectively.

3.2 Sub-vector quantization

Vector quantization (VQ) [20] has shown high efficiency for representing Gaussian attributes, capitalizing on their inherently vectorized structure and strong global coherence across an entire scene. However, to maintain high fidelity, a large codebook size is required, inevitably resulting in substantial computational overhead and increased training complexity [63] (Figure 3(a)). To address these issues, Residual Vector Quantization (R-VQ) [63] has been used as a hierarchical quantization strategy [32], progressively refining representations while reducing the size of each individual codebook. However, as shown in Figure 3(b), multiple code indices per attribute result in increased storage overhead, illustrating a tradeoff between reducing per-codebook complexity and increasing overall storage requirements.

To navigate this tradeoff, we propose Sub-Vector Quantization (SVQ), which partitions the attribute vector into multiple sub-vectors and applies vector quantization separately to each component (Figure 3(c)), motivated by Product Quantization [26]. By reducing the dimensionality of each quantized unit, SVQ allows for smaller codebooks and more efficient lookups, which can balance computational cost and storage efficiency while maintaining high fidelity. We can apply SVQ to an input vector $z \in \mathbb{R}^{ML}$, where M and L represent the total number of sub-vectors (partitions) and the sub-vector length, respectively. Each partition $m \in \{1, ..., M\}$ has an independent codebook $C^{(m)} \in \mathbb{R}^{B\times L}$, where B denotes the number of codewords per codebook. The codeword selection is based on the nearest match from $C^{(m)}$, with $C^{(m)}[j]$ representing the j-th codeword corresponding to the m-th sub-vector. More formally, SVQ-applied vector $\hat{z}$ can be formulated as follows,

$$\hat{z} := q(z; M) = \text{cat}(C^{(1)}[i_1], C^{(2)}[i_2], ..., C^{(M)}[i_M]), \tag{5}$$

$$i_m = \arg\min_j \|z_m - C^{(m)}[j]\|_2^2, \quad m \in \{1, ..., M\}, \tag{6}$$

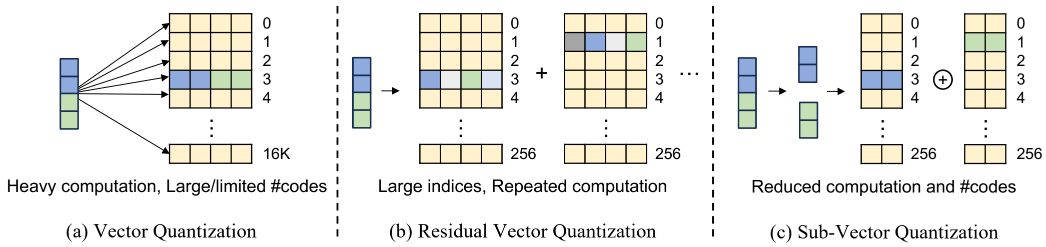

Figure 3: Conceptual diagram of (a) vector quantization, (b) residual vector quantization, and (c) sub-vector quantization. + and $\oplus$ denote the element-wise summation and the vector concatenation.

where $q(z; M)$ denotes applying SVQ with M sub-vectors and $i_m \in \{1, \ldots, B\}$ is the selected index of m-th sub-vector.

SVQ ensures significantly reduced computation with small codebooks compared to VQ. We apply SVQ to geometric attributes s_n, r_n, resulting in quantized vectors $\hat{s}_n, \hat{r}_n$, which are then used for 3DGS rendering. For the appearance features T_n, V_n, we first concatenate them and apply SVQ. The resulting quantized features are then split back into two components $\hat{T}_n, \hat{V}_n$, which replace T_n and V_n in Equations (3) and (4).

Although the reduced codebook size significantly decreases computational overhead compared to VQ, the process of updating both the indices and codes at every training iteration increases training time. Moreover, we observe that as training converges, the selected codebook indices remain largely unchanged. Therefore, we adopt a fine-tuning strategy in the final 1K iterations: after initializing with K-means, we freeze the indices and finetune only the codebook using the rendering loss, without introducing any additional losses. Since K-means initialization is completed within seconds due to the small codebooks, this approach adds minimal additional training time, unlike other methods that incur significant overhead.

3.3 Local distinctiveness for important scoring

OMG adopts importance scoring to identify essential Gaussians and retain a minimal number of them. Existing scoring-based pruning methods typically determine the importance of each Gaussian based on its blending weights (the values multiplied by c_k in Equation (1)) across training-view renderings. We use two factors as our baseline metric: (1) whether it has been the most dominant contributor for at least one ray [14, 15] and (2) its total blending weight contribution across all training rays [46, 19]. Formally, we define the base importance score $\bar{I}$ as:

$$\bar{I}_i = \begin{cases} \sum_{\rho=1}^{N^R} w_{i,\rho}, & \text{if } \exists \rho \in \{1, ..., N^R\} | w_{i,\rho} = \max_j w_{j,\rho}, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $w_{i,\rho}$ represents the blending weight of Gaussian i for ray ρ and N^R is the number of total rays in training views.

While this score captures global importance, it does not account for redundancy among closely-positioned Gaussians. In cases where multiple Gaussians are located in close proximity, their blending weights tend to be highly similar, thus naively thresholding them can lead to two potential issues: (1) abrupt performance degradation when all similar Gaussians are simultaneously removed, and (2) redundancy when multiple Gaussians with near-identical contributions are retained.

To mitigate these issues, we propose incorporating a local distinctiveness metric into the importance computation. Specifically, we introduce an additional term that measures the similarity of the static appearance feature $T \in \mathbb{R}^{N \times D}$ between neighboring Gaussians, ensuring that locally distinct Gaussians show high importance. The final importance score is defined as:

$$I_i = \bar{I}_i \left(\frac{1}{D} \sum_{j \in \mathcal{N}_i^K} \|T_i - T_j\|_1 \right)^{\lambda}, \tag{8}$$

where $\mathcal{N}_i^K$ denotes the set of K-nearest neighbors of Gaussian i, and λ is a scaling factor that adjusts the sensitivity to appearance variation. As computing exact K-nearest neighbors for every Gaussian is

Figure 4: Qualitative results of OMG compared to 3DGS and LocoGS with COLMAP initialization. We provide per-image rendering PSNR with storage requirements for each scene.

computationally expensive, we approximate neighbor selection by sorting Gaussians in Morton order and selecting Gaussians with adjacent indices as their local neighbors. We remove low-importance Gaussians using CDF-based thresholding [33] with a threshold τ.

4 Experiment

4.1 Implementation details

Following the previous works, we evaluated our approach on three real-world datasets, Mip-NeRF 360 [2], Tanks&Temples [31], and Deep Blending [23]. Our model is implemented upon Mini-Splatting [14], one of the methods achieving high performance with a small number of Gaussians. We have conducted simple post-processings after training: 1) Applying 16-bit quantization to the position and compressing with G-PCC [51]. 2) Huffman encoding [25] to SVQ indices. 3) Storing all the components into a single file with LZMA [1] compression. We provide five OMG variants (XS, X, M, L, XL), adjusting storage requirements. The only factor controlling the storage is the CDF-based threshold value τ of Gaussian importance, which is set to 0.96, 0.98, 0.99, 0.999, and 0.9999 for each variant, respectively. Further implementation details are provided in the supplementary materials.

4.2 Performance evaluation

Compression performance. Tables 1 and 2 compare the performance of OMG against various baseline methods on the Mip-NeRF 360, Tanks & Temples, and DeepBlending datasets. OMG consistently shows the smallest storage requirements while maintaining high performance across all datasets, achieving state-of-the-art (SOTA) results. Notably, on the Mip-NeRF 360 dataset, OMG-XS achieves nearly a 50% reduction in storage compared to the small variant of LocoGS [53], the previous SOTA compression method, while retaining PSNR and SSIM. With over 30% reduced storage, OMG-M outperforms LocoGS-S in all quality metrics. Moreover, OMG-XL surpasses LocoGS-L in all metrics, even though requiring less storage than LocoGS-S.

The qualitative results presented in Figure 4 also demonstrate the strong performance of OMG. Despite achieving over 100× compression compared to 3DGS, OMG maintains comparable visual quality. Especially, in the *bicycle* scene, OMG-XS achieves over 300× compression relative to 3DGS while accurately reconstructing details that 3DGS fails to represent, resulting in a blurry area (highlighted in red) in its rendering. This superiority can be attributed to the blur split technique of our baseline model, Mini-Splatting [14]. Despite reducing the number of Gaussians by an additional 20% compared to Mini-Splatting (Table 3), OMG-XS retains high visual fidelity, demonstrating its effectiveness in extreme compression scenarios.

Table 1: Quantitative results of OMG evaluated on the Mip-NeRF 360 dataset. Baseline results are sourced from the LocoGS [53] paper, where the rendering results were obtained using an NVIDIA RTX 3090 GPU. Our rendering performance was measured using the same GPU, with the values in parentheses obtained from an NVIDIA RTX 4090 GPU. LocoGS* refers to LocoGS initialized with COLMAP, instead of dense Nerfacto initialization. We highlight the results among compression methods by coloring the best, second-best, and third-best performances.

Method	Mip-NeRF 360				
	PSNR ↑	SSIM ↑	LPIPS ↓	Size(MB) ↓	FPS ↑
3DGS	27.44	0.813	0.218	822.6	127
Scaffold-GS [37]	27.66	0.812	0.223	187.3	122
Mini-Splatting [14]	27.39	0.822	0.216	119.5	(601)
CompGS [44]	27.04	0.804	0.243	22.93	236
Compact-3DGS [32]	26.95	0.797	0.244	26.31	143
C3DGS [45]	27.09	0.802	0.237	29.98	134
LightGaussian [13]	26.90	0.800	0.240	53.96	244
EAGLES [19]	27.10	0.807	0.234	59.49	155
SOG [42]	27.01	0.800	0.226	43.77	134
HAC [8]	27.49	0.807	0.236	16.95	110
LocoGS-S [53]	27.04	0.806	0.232	7.90	310
LocoGS-L [53]	27.33	0.814	0.219	13.89	270
LocoGS* (COLMAP)	27.09	0.798	0.250	7.96	(396)
	27.37	0.807	0.236	15.10	(325)
OMG-XS	27.06	0.807	0.243	4.06	350 (612)
OMG-M	27.21	0.814	0.229	5.31	298 (511)
OMG-XL	27.34	0.819	0.218	6.82	251 (416)

Table 2: Quantitative results of OMG evaluated on the Tanks&Temples and Deep Blending datasets. Baseline results are sourced from the LocoGS [53] paper, where the rendering results were obtained using an NVIDIA RTX 3090 GPU. Our rendering performance was measured using the same GPU, with the values in parentheses obtained from an NVIDIA RTX 4090 GPU.

Method	Tank&Temples					Deep Blending				
	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓	FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓	FPS ↑
3DGS [28]	23.67	0.844	0.179	452.4	175	29.48	0.900	0.246	692.5	134
Scaffold-GS [37]	24.11	0.855	0.165	154.3	109	30.28	0.907	0.243	121.2	194
Mini-Splatting [14]	23.41	0.846	0.180	67.6	(1095)	30.04	0.910	0.241	124.9	(902)
CompGS [44]	23.29	0.835	0.201	14.23	329	29.89	0.907	0.253	15.15	301
Compact-3DGS [32]	23.33	0.831	0.202	18.97	199	29.71	0.901	0.257	21.75	184
C3DGS [45]	23.52	0.837	0.188	18.58	166	29.53	0.899	0.254	24.96	143
LightGaussian [13]	23.32	0.829	0.204	29.94	379	29.12	0.895	0.262	45.25	287
EAGLES [19]	23.14	0.833	0.203	30.18	244	29.72	0.906	0.249	54.45	137
SOG [42]	23.54	0.833	0.188	24.42	222	29.21	0.891	0.271	19.32	224
HAC [8]	24.08	0.846	0.186	8.42	129	29.99	0.902	0.268	4.51	235
LocoGS-S [53]	23.63	0.847	0.169	6.59	333	30.06	0.904	0.249	7.64	334
LocoGS-L [53]	23.84	0.852	0.161	12.34	311	30.11	0.906	0.243	13.38	297
OMG-M	23.52	0.842	0.189	3.22	555 (887)	29.77	0.908	0.253	4.34	524 (894)
OMG-L	23.60	0.846	0.181	3.93	478 (770)	29.88	0.910	0.247	5.21	479 (810)

Computational efficiency. OMG achieves remarkable efficiency alongside high performance. As shown in Table 3, OMG shows superior scene fidelity with significantly fewer Gaussian primitives compared to LocoGS. This reduction results in substantial rendering speed improvements of 13%, 67%, and 57% for the Mip-NeRF 360, Tank&Temples, and Deep Blending datasets (Tables 1 and 2), respectively, compared to LocoGS, highlighting its potential for real-time rendering on low-capacity devices. Furthermore, OMG accelerates training speed. The substantial improvement over LocoGS can be attributed to two key factors: the reduced number of Gaussians and the absence of a large neural field. By efficiently exploiting coarse spatial information through a tiny MLP, OMG achieves high computational efficiency.

4.3 Ablation study

Local distinctiveness (LD) scoring. OMG improves Gaussian pruning by incorporating LD scoring into the importance estimation. Without attribute compression, LD scoring enables high

Table 3: Efficiency comparison of OMG variants evaluated on the Mip-NeRF 360 dataset. We present training time, the number of Gaussians, and the storage requirement with rendering quality.

Method	Training	#Gauss	Size	PSNR	SSIM	LPIPS
Mini-Splatting	19m 25s	531K	119.5	27.39	0.822	0.216
LocoGS-S	≈1h	1.09M	7.9	27.04	0.806	0.232
LocoGS-L		1.32M	13.89	27.33	0.814	0.219
OMG-XS	20m 15s	427K	4.06	27.06	0.807	0.243
OMG-S	20m 57s	501K	4.75	27.14	0.811	0.235
OMG-M	21m 10s	563K	5.31	27.21	0.814	0.229
OMG-L	21m 32s	696K	6.52	27.28	0.818	0.220
OMG-XL	22m 26s	727K	6.82	27.34	0.819	0.218

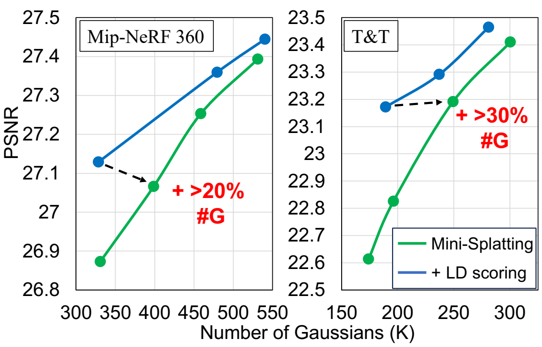

Figure 5: Evaluation without attribute compression.

Table 4: Ablation study of OMG using the Mip-NeRF 360 dataset.

Method	PSNR	SSIM	LPIPS	#Gauss	Size
OMG-M	27.21	0.814	0.229	0.56M	5.31
w/o Space feature	26.96	0.811	0.232	0.59M	5.58
w/o LD scoring	27.09	0.813	0.230	0.57M	5.36
w/o Both	26.81	0.809	0.234	0.59M	5.59
w/o SVQ	27.26	0.817	0.226	0.56M	26.1
OMG-XS	27.06	0.807	0.243	0.43M	4.06
w/o Space feature	26.85	0.804	0.246	0.44M	4.17
w/o LD scoring	26.83	0.804	0.246	0.43M	4.12
w/o Both	26.52	0.798	0.252	0.45M	4.24
w/o SVQ	27.06	0.809	0.241	0.43M	19.8

rendering quality with extremely reduced Gaussians, achieving similar performance compared to Mini-Splatting that use 20–30% more Gaussians, as shown in Figure 5. When applying attribute compression (Table 3), OMG with LD scoring still outperforms, leading to a significant performance improvement with a similar number of Gaussians. This effect becomes even more pronounced when the target Gaussian number is lower, demonstrating that LD scoring provides an effective approach for further reducing a sparse set of Gaussians.

OMG architecture. OMG leverages a highly compact neural field to capture coarse spatial information while reducing the number of learnable parameters per Gaussian. Table 4 validates the contribution of this space feature. Although the total number of Gaussians slightly increases, performance significantly degrades. The absence of spatial information introduces instability in attribute learning, hindering effective importance scoring. This trend is consistently observed in both our small and medium models, highlighting the effectiveness of the space feature despite its minimal parameter overhead. Furthermore, when both the space feature and LD scoring are removed, the model experiences the most substantial performance drop. This indicates that the two contributions are orthogonal, independently contributing to model efficiency and performance.

Sub-vector quantization. Table 4 shows the effectiveness of SVQ in terms of performance-storage efficiency. In addition, we replaced it with VQ and RVQ while keeping the proposed clever training strategy: K-means clustering is performed once prior to the final 1K iterations, after which only the codebook is updated. As shown in Table 5, VQ incurs a 13–18$\times$ increase in codebook initialization time compared to SVQ, resulting in significantly higher training overhead due to the need for large codebooks (2^{14} entries per scale, rotation, and two appearance features) to ensure accurate representation. Nevertheless, VQ leads to lower rendering quality and/or increased storage costs across variants. In contrast, RVQ achieves slightly faster codebook initialization than SVQ but performs poorly in terms of rate-distortion efficiency, yielding higher storage requirements and lower rendering quality. These results demonstrate that SVQ outperforms both VQ and RVQ in representing per-Gaussian attributes efficiently.

Generalization ability. The OMG attribute representation is applicable to all 3DGS-based methods, as it flexibly represents attributes with spatial locality (via space features and SVQ) as well as those without (using SVQ alone). To validate this generalizability, we applied OMG to 3DGS-MCMC [30], a method well-known for effective densification. As reported in Table 6, OMG preserves the original performance and reduce storage storage requirements, demonstrating its broader applicability.

Table 5: Ablation study of SVQ on the Mip-NeRF 360 dataset. We replace SVQ with either VQ or RVQ. 'CB Init' denotes the time (in seconds) required for codebook initialization.

Method	OMG-XS			OMG-M			OMG-XL		
Metric	CB Init↓	Size↓	PSNR↑	CB Init↓	Size↓	PSNR↑	CB Init↓	Size↓	PSNR↑
OMG	4.8	4.06	27.06	7.6	5.31	27.21	7.9	6.82	27.34
SVQ→VQ	86.4	4.21	27.06	100.9	5.36	27.18	117.4	6.70	27.27
SVQ→RVQ	4.0	4.19	27.01	5.5	5.52	27.13	6.9	7.10	27.25

Table 6: Results of applying OMG representation to 3DGS-MCMC [30] on the Mip-NeRF dataset.

Method	#Gauss	PSNR	SSIM	LPIPS	Size	#Gauss	PSNR	SSIM	LPIPS	Size
MCMC [30]	500K	27.42	0.807	0.248	115 MB	1M	27.83	0.823	0.221	230 MB
MCMC+OMG	500K	27.21	0.797	0.256	5.1 MB	1M	27.63	0.813	0.227	10.0 MB

Table 7: Performance evaluation using Zip-NeRF [3] dataset. * indicates results reported in the SMERF [12] paper, where 3DGS hyperparameters were tuned for higher performance. We train and report results for 3DGS and Mini-Splatting using their default settings.

Method	PSNR	SSIM	LPIPS	#Gauss	Size
Zip-NeRF* [3]	27.37	0.836	0.305	-	607 MB
SMERF* [12]	27.28	0.829	0.350	-	4.1 GB
3DGS* [28]	25.50	0.810	0.369	-	212 MB
3DGS [28]	25.16	0.813	0.358	934K	210 MB
Mini-Splatting [14]	24.57	0.802	0.370	337K	75.7 MB
MCMC [30]	26.37	0.838	0.318	3M	675 MB
MCMC+OMG	26.69	0.839	0.310	3M	29.0 MB

5 Limitations and broader applicability

Baseline dependancy. OMG builds upon 3DGS and Mini-Splatting with the goal of preserving original rendering quality under compact representations. However, its applicability remains inherently bounded by the capabilities of these baselines. For instance, 3DGS and subsequent approaches exhibit clear limitations in capturing the complexity of larger-scale scenes such as the Zip-NeRF [3] dataset. As shown in Table 7, both 3DGS and Mini-Splatting fail to densify a sufficient number of Gaussians, leading to degraded performance. Consequently, OMG inherits these limitations, restricting its applicability in such scenarios.

Broader applicability. On the other hand, as demonstrated in Table 6, OMG can be applied in a generalizable manner across different baseline methods. Based on this strength, we tune 3DGS-MCMC for large-scale scenes and incorporate the OMG representation, achieving superior performance while drastically reducing storage requirements (Table 7). More broadly, when integrated with ongoing research across diverse domains, OMG has the potential to provide a highly general and efficient representation.

6 Conclusion

In this paper, we proposed Optimized Minimal Gaussians (OMG), a novel framework that significantly reduces the number of Gaussian primitives while maximizing compressibility and maintaining high rendering quality. By effectively identifying and preserving locally distinct Gaussians, OMG minimizes the redundancy of Gaussians with minimal loss of visual fidelity. Furthermore, our compact and precise attribute representation, combined with sub-vector quantization, enables efficient exploitation of both continuity and irregularity, ensuring high efficiency. Experimental results demonstrate that OMG reduces storage requirements by nearly 50% compared to the previous state-of-the-art method while allowing over 600 FPS rendering performance. OMG sets a new benchmark for highly efficient 3D scene representations, paving the way for future advancements in real-time rendering on resource-constrained devices.

Acknowledgements

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grants (RS-2019-II190421, RS-2020-II201821, RS-2021-II212052, RS-2021-II212068, RS-2024-00457882, RS-2025-02217613, RS-2025-10692981, RS-2025-25442569) funded by the Korean government (MSIT); the Technology Innovation Program (RS-2023-00235718, 23040-15FC) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea) (1415187505); Samsung Research Funding & Incubation Center of Samsung Electronics (SRFC-IT2401-01); a grant of the Korea-US Collaborative Research Fund (KUCRF), funded by the Ministry of Science and ICT and Ministry of Health & Welfare, Republic of Korea (RS-2024-00468417).

References

[1] Lzma compression algorithm. `https://www.7-zip.org/sdk.html`.

[2] 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.

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022.

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, 2022.

[7] Yihang Chen, Qianyi Wu, Mehrtash Harandi, and Jianfei Cai. How far can we compress instant-ngp-based nerf? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20321–20330, 2024.

[8] 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, 2024.

[9] Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3d gaussian splatting compression. In *International Conference on Learning Representations*, 2025.

[10] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac++: Towards 100x compression of 3d gaussian splatting. *arXiv preprint arXiv:2501.12255*, 2025.

[11] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21401–21412, 2024.

[12] Daniel Duckworth, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T Barron. Smerf: Streamable memory efficient radiance fields for real-time large-scene exploration. *ACM Transactions on Graphics (TOG)*, 43(4), 2024.

[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. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 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, 2024.

[15] Guangchi Fang and Bing Wang. Mini-splatting2: Building 360 scenes within minutes via aggressive gaussian densification. *arXiv preprint arXiv:2411.12788*, 2024.

[16] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[17] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[18] Quankai Gao, Qiangeng Xu, Hao Su, Ulrich Neumann, and Zexiang Xu. Strivec: Sparse tri-vector radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17569–17579, 2023.

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

[20] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[21] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024.

[22] Kang Han and Wei Xiang. Multiscale tensor decomposition and rendering equation encoding for view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4232–4241, 2023.

[23] 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.

[24] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH*, 2024.

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

[26] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

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

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[29] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4), 2024.

[30] 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.

[31] 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.

[32] 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 (CVPR)*, pages 21719–21728, 2024.

[33] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Liefeng Bo. Compressing volumetric radiance fields to 1 mb. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4222–4231, 2023.

[34] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, pages 15651–15663, 2020.

[35] 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.

[36] Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks. *arXiv preprint arXiv:2412.20522*, 2024.

[37] 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.

[38] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *International Conference on 3D Vision (3DV)*, pages 800–809, 2024.

[39] 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.

[40] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, page 405–421, 2020.

[41] Arthur Moreau, Jifei Song, Helisa Dhamo, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. Human gaussian splatting: Real-time rendering of animatable avatars. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 788–798, 2024.

[42] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. In *European Conference on Computer Vision*, pages 18–34, 2024.

[43] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), 2022.

[44] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster gaussian splatting with vector quantization. In *European Conference on Computer Vision*, pages 330–349, 2024.

[45] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10349–10358, 2024.

[46] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv preprint arXiv:2403.13806*, 2024.

[47] 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), 2024.

[48] 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.

[49] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024.

[50] Daniel Rho, Byeonghyeon Lee, Seungtae Nam, Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Masked wavelet representation for compact neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20680–20690, 2023.

[51] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2018.

[52] Seungjoo Shin and Jaesik Park. Binary radiance fields. In *Advances in Neural Information Processing Systems*, volume 36, pages 55919–55931, 2023.

[53] Seungjoo Shin, Jaesik Park, and Sunghyun Cho. Locality-aware gaussian compression for fast and high-quality rendering. In *International Conference on Learning Representations*, 2025.

[54] 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.

[55] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, 2022.

[56] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *International Conference on Learning Representations*, 2024.

[57] Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. End-to-end rate-distortion optimized 3d gaussian representation. In *European Conference on Computer Vision*, pages 76–92. Springer, 2024.

[58] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024.

[59] Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex Kot, and Bihan Wen. Contextgs : Compact 3d gaussian splatting with anchor level context model. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 51532–51551, 2024.

[60] 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, 2024.

[61] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[63] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

[64] Yangming Zhang, Wenqi Jia, Wei Niu, and Miao Yin. Gaussianspa: An" optimizing-sparsifying" simplification framework for compact and high-quality 3d gaussian splatting. *arXiv preprint arXiv:2411.06019*, 2024.

[65] 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, 2025.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contribution, Optimized Minimal Gaussians (OMG) representation, which reduces storage requirements and enables faster rendering compared to existing 3DGS methods.

   Guidelines:

   - The answer NA 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 NA 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 of our method in Section 5.

   Guidelines:

   - The answer NA 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: [NA]

Justification: The paper does not present formal theoretical results or proofs.

Guidelines:

- The answer NA 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 sufficient details for reproducing the main experimental results, including descriptions of the datasets used, evaluation metrics, and implementation details.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The experiments are conducted using open datasets, and the paper provides detailed implementation descriptions. Additionally, the source code is attached in the supplementary material to ensure reproducibility.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies training and testing details, including datasets, hyperparameters, and optimizer settings, necessary to understand the results.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Error bars are not provided, following the common practice in this field where experimental results typically do not vary significantly across runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper provides information on the compute resources used, including GPU models (e.g., NVIDIA RTX 3090 and 4090), and reports training times and rendering speeds achieved.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research adheres to ethical guidelines, with no indication of unethical practices or violations of the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA 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: [NA]

    Justification: There is no societal impact of the work performed. This paper is focused on optimizing the efficiency of an existing framework.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not release models or data with high risk for misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper references existing datasets and methods used, respecting their licenses and terms of use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The new assets introduced, including the OMG codebase, are well documented and provided alongside the paper for reproducibility.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The research does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The research does not involve the use of large language models as a core component of the methodology.

    Guidelines:

    - The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Difficulty in compressing small set of Gaussians

To assess the challenge of compressing a small number of Gaussians, we integrate Mini-Splatting [14] with existing methods [32, 53]. We trained LocoGS [53] with Mini-Splatting-based densification in an end-to-end manner. For Compact-3DGS [32], however, we observe that training it directly with Mini-Splatting-based pruning fails to converge, due to inaccurate attribute representations during training. Rather, we halt Mini-Splatting training early at 20K iterations, where the pruning process is ended, and integrate Compact-3DGS representation in the subsequent 10K iterations.

As shown in Table 8, the integration of Mini-Splatting results in a significant degradation in visual quality across both methods, primarily caused by imprecise attribute representations for irregular and sparse Gaussians. In contrast, OMG preserves high visual fidelity with drastically reduced storage by through its novel attribute representation.

Table 8: Performance evaluation compared to Compact-3DGS [32] and LocoGS [53] with the reduced number of Gaussians.

Method	PSNR	SSIM	LPIPS	#G	Size
Mini-Splatting (MS)	27.39	0.822	0.216	531K	120 MB
Compact-3DGS	27.03	0.797	0.247	1.39M	29.1 MB
MS + Compact-3DGS	26.85	0.807	0.234	525K	14.8 MB
LocoGS-S	27.04	0.806	0.232	1.09M	7.9 MB
MS + LocoGS-S	26.52	0.789	0.257	537K	6.2 MB
LocoGS-L	27.33	0.814	0.219	1.32M	13.9 MB
MS + LocoGS-L	26.76	0.799	0.247	525K	11.3 MB

B KNN approximation using Morton order

Table 9 presents a comparison between Morton-order-based approximation and KNN for LD scoring. The performance is nearly identical, demonstrating that Morton order effectively approximates KNN while achieving a 146$\times$ speedup, even with the smallest model. While using KNN is reasonable in our current setup (LD scoring once at 20K with fewer than 1M Gaussians), our decision to adopt Morton order is motivated by the need for scalability and generalizability. In more complex or larger-scale scenes, the number of Gaussians can increase significantly during earlier training stages. In such cases, KNN becomes impractical due to its high memory usage and computational complexity of $O(N^2)$. Morton order, by contrast, provides a well-approximated yet efficient alternative that scales better with the number of primitives.

Table 9: Performance evaluation of KNN approximation on the Mip-NeRF 360 dataset.

Method	PSNR	SSIM	LPIPS	Time (ms)
OMG-XS	27.06	0.807	0.243	4100
Morton $\rightarrow$ KNN	27.07	0.808	0.241	28

C Rendering speed on weaker devices

We have measured the FPS of our method on an NVIDIA RTX 4090 and a lower-end GPU, NVIDIA GTX 1080 Ti. As shown in Table 10, OMG achieves substantially higher FPS on both devices compared to LocoGS, primarily due to its drastically reduced number of Gaussians. Especially, OMG-XS show 1.5× faster rendering on the GTX 1080Ti while requiring only half the storage and preserving high visual quality. These results indicate that OMG's efficiency directly extends to practical real-time applications, enabling deployment on lower-end GPUs.

Table 10: Rendering FPS evaluated on the RTX 4090 and GTX 1080Ti, compared to LocoGS (with COLMAP initialization).

Method	4090	1080Ti	PSNR	SSIM	LPIPS	#Gauss	Size (MB)
LocoGS	396	73	27.09	0.798	0.250	1.04M	7.96
OMG-XS	612	106	27.06	0.807	0.243	0.43M	4.06
LocoGS	325	59	27.37	0.807	0.236	1.44M	15.1
OMG-XL	416	77	27.34	0.819	0.218	0.73M	6.82

Table 11: Ablation study on the post-processing methods applied in OMG.

G-PCC	Huffman	OMG-XS	OMG-S	OMG-M	OMG-L	OMG-XL
-	-	5.82	6.83	7.66	9.47	9.89
✓	-	4.30	5.04	5.64	6.92	7.25
-	✓	5.58	6.54	7.33	9.08	9.46
✓	✓	**4.06**	**4.75**	**5.31**	**6.52**	**6.82**

D Implementation details

D.1 Mip-NeRF 360 dataset

All experiments were conducted using an NVIDIA RTX 4090. Our method was implemented within the Mini-Splatting [14] framework and trained for 30K iterations. At the 20K iteration simplification process, local distinctiveness scoring was incorporated where the factor λ was set to 2. The dimension of appearance features D was set to 3. Scale and rotation were trained from the initial training, while appearance features were introduced at 15K iterations. At this stage, the static features were initialized using the spherical harmonics DC coefficients trained until 15K iterations, whereas view-dependent features were initialized as zero vectors.

From 29K iterations (last 1K iterations), SVQ (Sub-Vector Quantization) was applied to per-Gaussian features. As mentioned in the paper, to enhance training efficiency, K-means clustering was performed once. The assigned indices based on K-means were fixed, and only the codebooks were optimized for the remaining 1K iterations. For SVQ, different bit allocations were assigned.

- Scale: length 1, 2^6 codes for each sub-vector
- Rotation: length 2, 2^9 codes for each sub-vector
- Appearance features: length 2, 2^{10} codes for each sub-vector

The length 1 SVQ applied to scale can be interpreted as scalar quantization, dynamically learning the quantization range with the codebooks. All codes in the codebook are stored with 16-FP precision.

This SVQ configuration was commonly applied across all variants from XS to XL. The model storage for each variant was determined only by the importance score threshold τ, which is used for simplification at the 20K iteration, set to 0.96, 0.98, 0.99, 0.999, and 0.999, respectively.

D.2 Zip-NeRF dataset

We used the 3DGS-MCMC [30] framework, and tuned it for the Zip-NeRF dataset, by training 3M Gaussians for 150K iterations with a densification interval of 1K iterations. Other hyperparameters are set to default values. For OMG training, the appearance features were introduced at 100K iterations, and from 140K iterations (last 10K iterations), SVQ was applied to per-Gaussian features. For SVQ, bit allocations were assigned as follows:

- Scale: length 1, 2^7 codes for each sub-vector
- Rotation: length 2, 2^{10} codes for each sub-vector
- Appearance features: length 2, 2^{11} codes for each sub-vector

Table 12: The average storage allocation for each component across OMG variants. 'Actual size' refers to the total size of a single file containing all components.

Attribute	OMG-XS	OMG-S	OMG-M	OMG-L	OMG-XL
Position	0.93	1.08	1.20	1.43	1.52
Scale	0.83	0.97	1.09	1.33	1.41
Rotation	0.87	1.02	1.15	1.40	1.49
Appearance	1.39	1.63	1.82	2.22	2.35
MLPs	0.03	0.03	0.03	0.03	0.03
Total	4.04	4.73	5.29	6.42	6.80
Actual size	4.06	4.75	5.31	6.52	6.82

E Effect of post-processings

As mentioned in the main paper, we applied the following two post-processing methods:

- Compressing the 16-bit quantized position with G-PCC [51].
- Huffman encoding [25] to SVQ indices and compressing the results with LZMA [1].

Both methods are applied losslessly, and we report the resulting storage changes in Table 11. When applied independently, G-PCC and Huffman encoding consistently reduce the total storage by 26-27% and 4-5% across all storage budgets, respectively. Applying both methods together also results in the overall storage reduction remaining consistent at approximately 30-32%.

F Storage analysis

We conducted experiments to analyze the storage requirements of OMG for representing each attribute, as shown in Table 12. Across all variants, OMG allocates approximately 20-25% of the total storage to position, scale, and rotation, while around 35% is dedicated to representing appearance attributes, including static and view-dependent color as well as opacity. The four MLPs for representing local continuity and aggregating appearance attributes exhibit negligible storage requirements, even without extra compression.

G Per-scene results

We report per-scene results in Table 13 (Mip-NeRF 360 [2]) and Table 14 (T&T [31] and DB [23]).

Table 13: Per-scene results evaluated on the Mip-NeRF 360 [2] dataset.

Method	Metric	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	Avg.
OMG-XS	PSNR	24.95	30.90	28.40	21.32	26.42	30.81	31.09	27.00	22.60	27.06
	SSIM	0.743	0.932	0.899	0.596	0.818	0.919	0.918	0.788	0.647	0.807
	LPIPS	0.276	0.202	0.206	0.368	0.190	0.137	0.208	0.247	0.357	0.243
	Train	18:03	20:30	24:44	19:18	18:02	23:45	20:30	17:49	19:40	20:15
	#Gauss	480772	263892	310056	543034	607254	356752	281236	523821	479520	427371
	Size	4.61	2.53	2.95	5.24	5.65	3.33	2.67	4.95	4.64	4.06
	FPS	682	648	433	616	615	498	648	708	658	612
OMG-S	PSNR	25.08	31.05	28.56	21.18	26.56	30.89	31.20	27.08	22.64	27.14
	SSIM	0.750	0.936	0.903	0.602	0.826	0.921	0.922	0.792	0.650	0.811
	LPIPS	0.264	0.195	0.199	0.358	0.177	0.132	0.201	0.239	0.347	0.235
	Train	19:01	21:09	25:19	20:13	18:41	24:12	21:38	18:29	19:55	20:57
	#Gauss	573126	310096	360930	633607	691441	412126	338884	619734	573425	501485
	Size	5.46	2.94	3.41	6.10	6.43	3.83	3.19	5.83	5.54	4.75
	FPS	601	585	401	555	556	462	620	601	588	552
OMG-M	PSNR	25.14	31.06	28.62	21.40	26.71	31.05	31.30	27.06	22.55	27.21
	SSIM	0.756	0.938	0.905	0.606	0.832	0.923	0.923	0.794	0.652	0.814
	LPIPS	0.256	0.190	0.195	0.351	0.169	0.129	0.198	0.233	0.339	0.229
	Train	18:58	21:01	25:44	20:35	18:51	24:18	22:14	18:31	20:22	21:10
	#Gauss	646191	350999	400442	708074	772338	454908	375520	704907	649157	562504
	Size	6.15	3.33	3.76	6.79	7.18	4.21	3.53	6.61	6.24	5.31
	FPS	562	536	371	510	522	440	566	566	525	511
OMG-L	PSNR	25.24	31.47	28.66	21.45	26.83	31.03	31.26	27.05	22.57	27.28
	SSIM	0.762	0.941	0.907	0.613	0.837	0.924	0.926	0.795	0.653	0.818
	LPIPS	0.241	0.183	0.189	0.338	0.160	0.126	0.191	0.226	0.329	0.220
	Train	19:25	21:16	26:06	20:50	19:14	24:20	22:05	19:22	21:14	21:32
	#Gauss	813561	463285	480133	859963	909961	524457	524457	869388	819435	696071
	Size	7.69	4.32	4.48	8.23	8.42	4.82	4.82	8.14	7.81	6.52
	FPS	476	492	332	422	422	405	539	468	414	441
OMG-XL	PSNR	25.22	31.51	28.78	21.52	26.93	31.15	31.25	27.00	22.69	27.34
	SSIM	0.764	0.942	0.908	0.614	0.839	0.925	0.926	0.796	0.655	0.819
	LPIPS	0.239	0.182	0.187	0.334	0.157	0.126	0.191	0.224	0.324	0.218
	Train	20:43	21:54	26:21	22:09	20:23	24:56	22:37	20:22	22:33	22:26
	#Gauss	864124	450246	507473	922061	953050	547636	493754	920589	885229	727129
	Size	8.15	4.22	4.72	8.81	8.82	5.02	4.58	8.59	8.44	6.82
	FPS	430	465	324	379	422	397	512	435	384	416

Table 14: Per-scene results evaluated on the Tank&Temples [31] and Deep Blending [23] datasets.

Method	Metric	Tank&Temples			Deep Blending		
		Train	Truck	Avg.	drjohnson	Playroom	Avg.
OMG-M	PSNR	21.78	25.25	23.52	29.37	30.18	29.77
	SSIM	0.806	0.878	0.842	0.905	0.910	0.908
	LPIPS	0.233	0.144	0.189	0.253	0.253	0.253
	Train	12:12	11:30	11:51	17:18	14:51	16:05
	#Gauss	303187	257649	330418	520385	404237	462311
	Size	2.95	3.49	3.22	4.87	3.82	4.34
	FPS	861	913	887	829	959	894
OMG-L	PSNR	21.85	25.36	23.60	29.44	30.32	29.88
	SSIM	0.811	0.881	0.846	0.907	0.912	0.910
	LPIPS	0.225	0.136	0.181	0.247	0.247	0.247
	Train	12:12	11:39	11:56	17:39	14:58	16:19
	#Gauss	369440	442359	405900	627868	485329	556599
	Size	3.58	4.28	3.93	5.86	4.55	5.21
	FPS	760	780	770	745	874	810