NeuralGS: Bridging Neural Fields and 3D Gaussian Splatting for Compact 3D Representations

Zhenyu Tang[1*], Chaoran Feng[1*], Xinhua Cheng[1,2], Wangbo Yu[1,2], Junwu Zhang[1], Yuan Liu[3], Xiaoxiao Long[3], Wenping Wang[4], Li Yuan[1,2†]

[1]Peking University, [2]Pengcheng Laboratory
[3]Hong Kong University of Science and Technology
[4]Texas A&M University

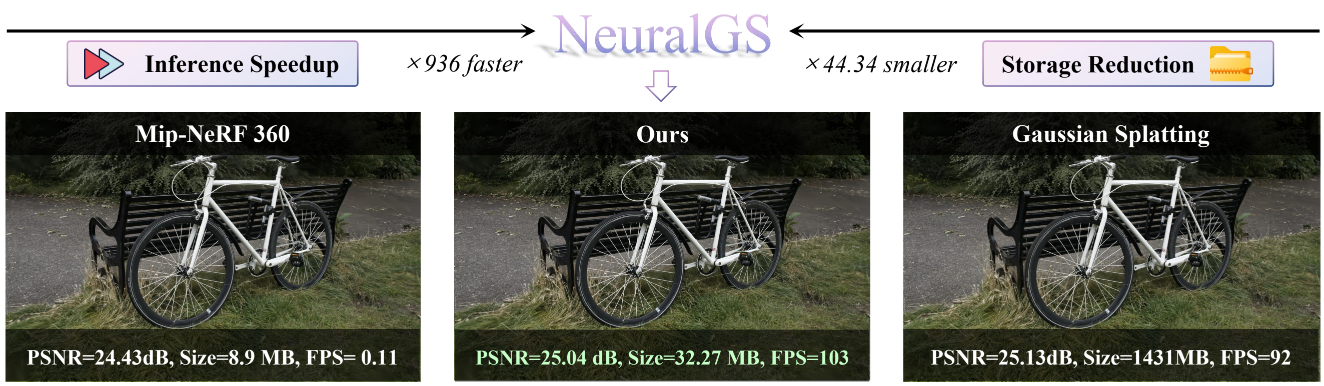


Figure 1. **NeuralGS** is a compact and rendering-efficient representation with small model sizes and fast rendering speed. NeRF-based methods like Mip-NeRF-360 [3] typically require minimal storage with slow rendering speeds while 3D Gaussian Splatting [16] (3DGS) methods achieve fast rendering but demand hundreds of megabytes storage. NeuralGS combines the compact neural fields with 3DGS by encoding 3D Gaussian attributes with neural fields, achieving significant model size reduction and real-time rendering speed.

Abstract

Recently, 3D Gaussian Splatting (3DGS) has gained popularity, demonstrating superior quality and rendering speed in novel view synthesis. However, 3DGS requires millions of 3D Gaussians, each with extensive associated attributes, resulting in significant storage and transmission costs. In contrast, neural fields like NeRF can represent complex 3D scenes with Multi-Layer Perceptron (MLP) neural networks using only a few megabytes. In this paper, we present a novel Gaussian compression method ***NeuralGS*** *that effectively adopts the neural field representation to encode the attributes of 3D Gaussians with MLPs, requiring a small storage size even for a large-scale scene. However, naively fitting the Gaussian attributes with an MLP network leads to severely degenerated quality. To address this, we adopt a clustering strategy and fit the Gaussians with multiple tiny MLPs for different clusters, based on importance scores of Gaussians as fitting weights. We validate our approach on multiple datasets, achieving a 38× average model size reduction without harming the visual quality.*

1. Introduction

Novel view synthesis (NVS) is a fundamental task in 3D vision, with substantial applications across fields such as virtual reality [6], augmented reality [45], and media generation [32]. This task aims to generate photo-realistic images of 3D scenes from novel views, given limited multi-view data. Neural radiance field (NeRF) [27] has gained significant attention as a 3D scene representation for its compact structure and exceptional capability to reconstruct large-scale scenes [2, 3, 21, 31, 36, 42], However, a persistent challenge hindering the widespread adoption of NeRF lies in the computational bottlenecks imposed by volumetric rendering [7], which limit the utilization in real scenes

*These authors contributed equally to this work.
†Corresponding author.

that require fast rendering speeds.

3D Gaussian Splatting (3DGS) [16] has emerged as an alternative representation, utilizing a point-based representation associated with 3D Gaussian attributes. Unlike the slow volume rendering of NeRFs, 3DGS utilizes a fast differentiable splatting technique, achieving exceptionally fast rendering speeds and promising image quality. However, employing point-based representations inherently leads to substantial storage demands, as millions of points and their attributes are stored independently, which significantly hinders the compactness of 3DGS as a scene representation.

Based on the above observations, we pose the question: *Can we combine the point-based rendering of 3DGS with the compact structure of NeRF to address the heavy storage demands of 3DGS and achieve a more compact representation?* A straightforward solution is to directly employ a multi-layer perceptron (MLP) neural network to map the positions of Gaussians to their attributes, which could represent these attributes with a compact neural field. However, fitting just a single MLP to represent Gaussian attributes leads to large fitting errors, severely degenerating the rendering quality of 3D scenes, because the Gaussians show strong spatial variations. Even nearby 3D Gaussians have totally different attributes, resulting in a significant difficulty in fitting them with a single MLP.

To address the aforementioned issues, we propose **NeuralGS**, a novel framework designed for the post-training compression of 3D Gaussians, which bridges 3D Gaussian splatting and neural radiance field for a compact and efficient 3D scene representation. We adopt three strategies to facilitate the effective encoding of 3D Gaussian attributes with neural fields as follows:

First, instead of fitting all attributes of all Gaussians equally, we compute the importance of each Gaussian according to their contributions to the renderings. Gaussians with low importance are first pruned to reduce the Gaussian numbers. Furthermore, the importance of Gaussians acts as weights in the fitting process, which ensures that important Gaussians are fitted with high accuracy.

Second, to reduce attribute variability among Gaussians, we cluster 3D Gaussians based on their attributes to preserve similarity among Gaussians within the same cluster. For different clusters, we use different tiny MLPs to fit their attributes. This clustering and fitting scheme leads to multiple tiny neural fields to represent the Gaussian attributes, significantly reducing the fitting errors and improving the compactness of 3D representation.

Third, we further fine-tune the learned NeuralGS representation with input images and propose a novel frequency loss to improve the reconstruction quality. We find that the MLPs often have difficulty in learning the high-frequency signals of Gaussian attributes. Thus, we incorporate a frequency loss, that puts more emphasis on the high-frequency

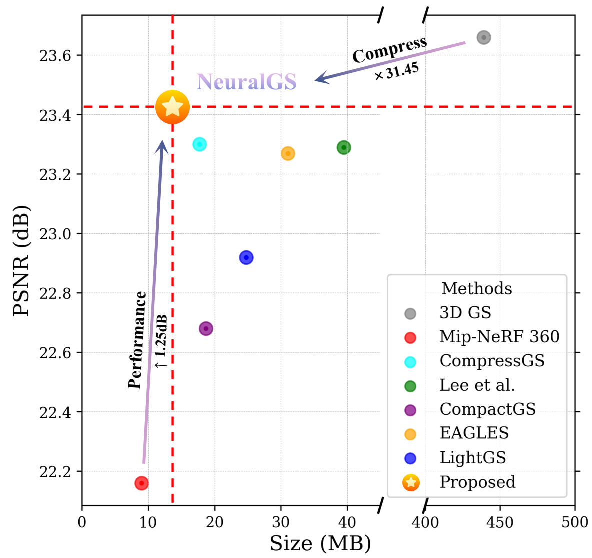


Figure 2. **Rendering quality vs. Model size.** We compare the proposed method with the existing Gaussian splatting methods [10, 11, 16, 18, 28, 30] on the *Tanks&Templates* dataset [17].

components of renderings, along with the original rendering loss in the fine-tuning process to recover these fine details.

In the end, our NeuralGS only need to store the positions of important Gaussians and the weights of the corresponding tiny MLPs for all clusters, substantially reducing storage requirements compared to the original 3DGS.

We conduct comprehensive experiments to evaluate our approach across a variety of datasets, achieving comparable quality as 3DGS and high compression ratio, such as about **39×** model size reduction on the Mip-NeRF360 dataset [3] and DeepBlending dataset [14]. We also compress 3DGS on the NeRF-Synthetic dataset [27] with only **1.7MB** of storage, while maintaining high rendering quality. In Figure 2, we achieve an optimal balance between rendering quality and model size for all compared methods. Additionally, by transmitting and decoding 3D Gaussians cluster by cluster, we can enable progressive loading of the 3D scenes.

2. Related Works

2.1. Novel View Synthesis

Neural radiance field (NeRF) [27] proposes to use multi-layer perceptron (MLPs) to represent a scene, and this compact representation has brought view synthesis quality to a new stage. However, NeRF-based methods [3, 12, 15, 19, 29, 31, 33] struggle to achieve real-time rendering speed in large-scale scenes, limiting their practical use. The idea of utilizing multiple MLPs is also explored by KiloNeRF [33] for efficient rendering. Recently, 3D Gaussian Splatting (3D-GS) [16] and its variants [20, 22, 24, 26, 35, 40, 41, 43], offer state-of-the-art scene reconstruction by utilizing an

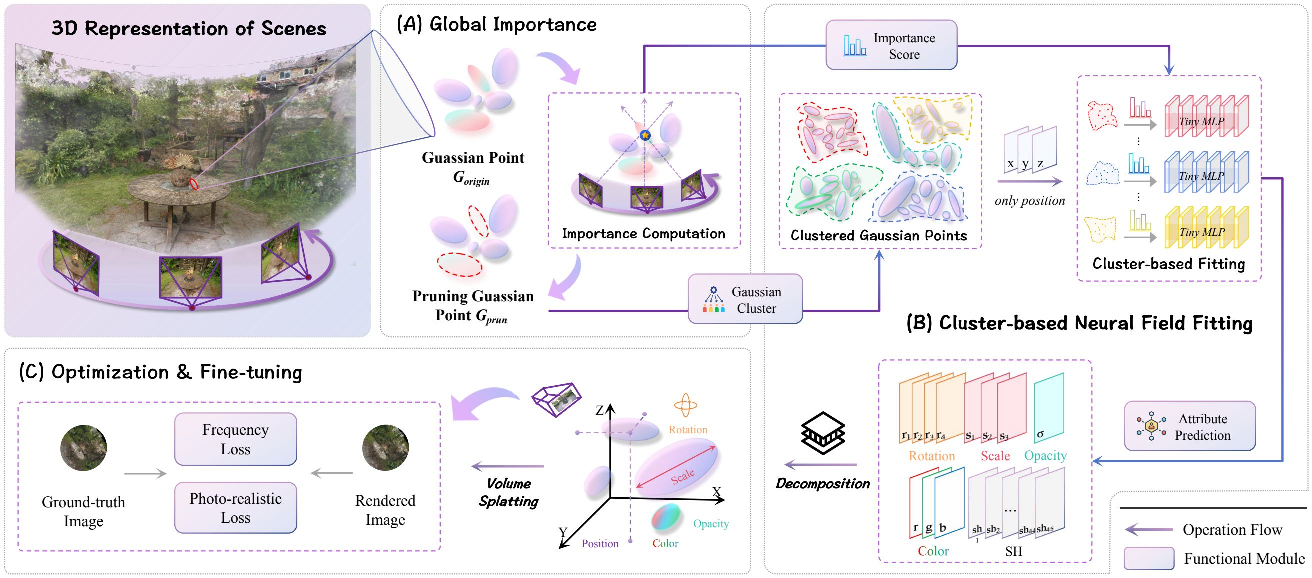


Figure 3. **The detailed architecture of our proposed NeuralGS**. (A) For each Gaussian GS_j in the scene, we first calculate its global importance score S_j (Eq.2) and prune unimportant Gaussians. (B) Next, we cluster the pruned Gaussians and use different tiny MLPs to fit Gaussian attributes of different clusters with the loss (Eq.4) using the importance score as weights. (C) Finally, we fine-tune the tiny MLPs of each cluster with photorealistic loss (Eq.5) and frequency loss (Eq.6) to restore quality.

optimized set of 3D Gaussians that can be rendered efficiently. Through the differentiable tiled rasterizer, 3D Gaussians are optimized during training to best fit the 3D scene.

2.2. Compression of 3D Gaussian Splatting

Although 3DGS achieves superior performance and high rendering speed compared to NeRF-based methods, it typically requires hundreds of megabytes to represent 3D Gaussian attributes, posing challenges for its practical application in large-scale scenes. Several existing works [1, 9, 10, 18, 30, 39] have made initial attempts to compress 3DGS models, primarily using pruning to reduce the number of 3D Gaussians, vector quantization to discretize Gaussian attributes into shared codebooks, and context-aware entropy encoding. Specifically, Lee et al. [18] introduced a novel volume-based masking strategy that effectively reduces the number of Gaussians without impacting performance. CompressGS [30] employs the sensitivity to compress both color and Gaussian parameters into compact codebooks while utilizing entropy coding to minimize statistical redundancies in the codebooks. LightGS [10] reduces the number of Gaussians through pruning and effectively minimizes the size of color attributes using a distillation mechanism. On the other hand, CompactGS [28] proposed a 2D grid-based representation to compress the attributes, while the three works [4, 5, 25] utilize an anchor-based Gaussian splatting representation to model the relationships among Gaussians. In addition, the concurrent work [38] employs a triplane representation for Gaussian attributes. In contrast, we attempt to employ more compact neural fields to encode Gaussian attributes with tiny MLPs.

3. Method

3.1. Preliminaries

3D Gaussian Splatting. 3DGS [16] represents the scene with a series of sparse 3D Gaussians. Each Gaussian is parameterized by a 3D covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{3\times3}$ and location $\mu \in \mathbb{R}^3$:

$$\mathbf{G}(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)}, \tag{1}$$

where the covariance matrix $\mathbf{\Sigma}$ can be further factorized into a scaling matrix $\mathbf{S} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$, represented as $\mathbf{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top$. To render an image on a specific camera pose, the covariance matrix in camera coordinates, denoted as $\mathbf{\Sigma}' = \mathbf{JW\Sigma W}^\top\mathbf{J}^\top$, where $\mathbf{J}$ represents the Jacobian of the affine approximation of the projective transformation, and $\mathbf{W}$ denotes the view transformation matrix. Subsequently, the color of each pixel on the image plane is determined by blending N Gaussians arranged in accordance with their respective depths, calculated as $\mathbf{C} = \sum_{i=1}^{N} \mathbf{T}_i\alpha_i\mathbf{c}_i$, where α_i is computed from a 2D Gaussian with covariance $\mathbf{\Sigma}$ multiplied by the optimizable opacity of the corresponding 3D Gaussian.

3.2. Overview

We propose a simple yet effective framework **NeuralGS**, which adopts the neural field representation to encode the

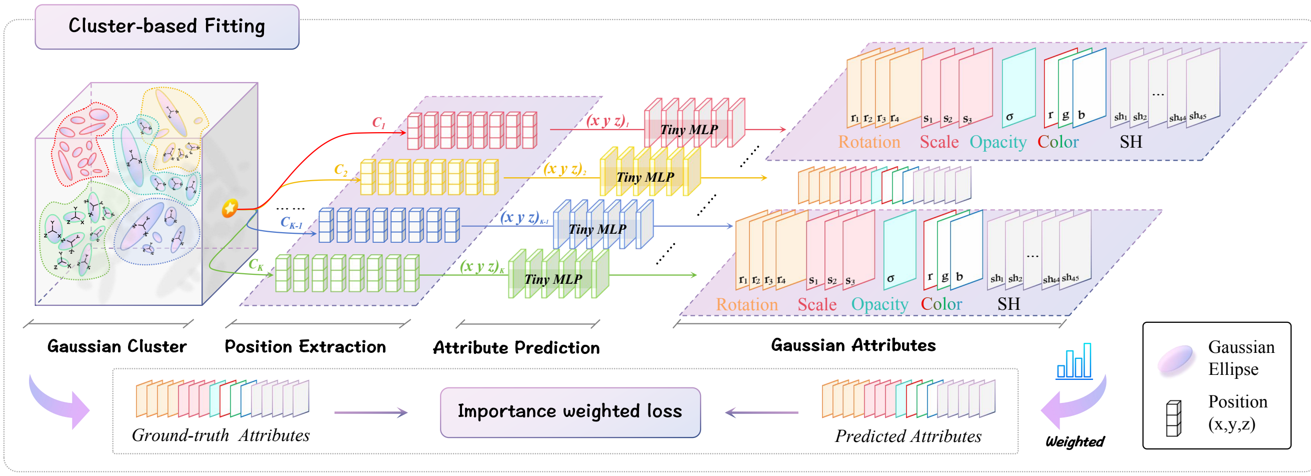


Figure 4. **Details of Cluster-based Neural Field Fitting.** The positions of the 3D Gaussians within each cluster are fed into the corresponding tiny MLP to fit the attributes with the importance weighted loss. During rendering, the predicted outputs are then split into the respective attributes of the Gaussians, i.e., rotation, scale, opacity, color, and SH coefficients.

attributes of 3D Gaussians with MLPs to enable a compact 3D representation. Specifically, as shown in Figure 3, we use a designed criterion to assess the importance of each Gaussian, allowing us to prune Gaussians that have minimal impact on renderings. To reduce variations among Gaussians, we cluster these 3D Gaussians based on their attributes, ensuring similarity within each cluster. Then, each cluster is then assigned a tiny MLP that fits the attributes of its 3D Gaussians. Given the varying contributions of each Gaussian to the renderings, we apply Gaussian's importances as the fitting weights in the MLP fitting. We also incorporate a fine-tuning stage along with frequency loss to restore quality and preserve high-frequency details.

3.3. Global Importance

Importance computation. Each Gaussian in the 3D scene contributes differently to the final renderings in 3DGS [16]. To quantify this, we define a global importance score for each Gaussian, representing its contribution to the rendering result. Inspired by [10], the importance score can be calculated based on each Gaussian's contribution to every pixel p_i across all training views. We use the criterion $\mathbb{1}(\mathrm{GS}_j, p_i)$ to determine whether a Gaussian GS_j overlaps with pixel p_i after projection onto the 2D plane. At last, we can iterate over all training pixels and sum up the accumulated opacity of GS_j, denoted as $\alpha_k \prod_{l=1}^{k-1}(1-\alpha_l)$, to compute each Gaussian's contribution to the rendering result. Here, k is the index of the Gaussian GS_j in the depth ordering for pixel p_i. This importance score can be further refined by incorporating the 3D Gaussian's normalized volume V_{norm}. Finally, the global importance score can be expressed as:

$$S_j = \sum_{i=1}^{MHW} \mathbb{1}(\mathrm{GS}_j, p_i) \cdot (V_{\text{norm}})^{\beta} \cdot \alpha_k \prod_{l=1}^{k-1}(1-\alpha_l), \quad (2)$$

$$V_{\text{norm}} = \min\left(\max\left(\frac{V}{V_{\text{max90}}}, 0\right), 1\right). \quad (3)$$

Here, S, M, H, and W represent the importance score, the number of training views, the image height, and the image width, respectively. V_{max90} denotes the 90% largest volume of all sorted Gaussians, and β is the hyperparameter to enhance the score's flexibility.

Importance-based pruning. Thus, we rank each Gaussian in the 3D scene based on its importance score, allowing us to prune Gaussians with lower contributions to the renderings, thereby reducing the total number of Gaussians. Additionally, the importance scores of the 3D Gaussians can be used as weights in the subsequent fitting process, ensuring that important Gaussians are fitted with higher accuracy.

3.4. Cluster-based Neural Field Fitting

Due to the substantial differences among 3D Gaussians, using only a single MLP to encode Gaussian attributes may lead to significant fitting errors, severely degrading the rendering quality of the 3D scene. To address this, we cluster Gaussians by their attributes to ensure similarity within each cluster. We then assign distinct tiny MLPs to different clusters as shown in Figure 4, effectively reducing the fitting errors. Additionally, we apply importance scores of the Gaussians as fitting weights, enhancing the accuracy for high-importance Gaussians.

Gaussian clustering. Specially, we employ the K-Means algorithm [23] to cluster the 3D Gaussians into

K clusters, denoted as $C_1, C_2, \ldots, C_K$, aiming to reduce Gaussians variability within each cluster. Given the significant distributional differences across attributes, we first normalize each attribute to the range $[-1, 1]$ by computing its maximum and minimum values to prevent over-reliance on certain attributes during clustering. To accelerate K-Means clustering, we employ a batched strategy and refine the clustering results with multiple iterations.

Neural fields. After assigning a cluster index to each 3D Gaussian, we use distinct tiny MLPs for each cluster to fit the corresponding Gaussian attributes within the cluster. To mitigate fitting errors caused by attribute distribution differences, we also utilize the normalized Gaussian attributes during the fitting process. Each tiny MLP consists of five layers with positional encoding, followed by a tanh activation function [8] that aligns with the normalized attribute range. Considering the independence of different clusters, we employ multiprocessing to fit the Gaussian attributes of each cluster in parallel, further reducing training time.

Importance weighted loss. We apply mean squared error(MSE) loss during the fitting process of Gaussian attributes. Recognizing that each Gaussian contributes differently to the final result, we use the calculated importance scores as fitting weights. This approach ensures that Gaussians with higher importance are fitted more accurately. Our loss function is defined as:

$$Loss = \frac{1}{\sum_{j\in\mathcal{P}} S_j} \sum_{j\in\mathcal{P}} S_j \cdot \left\| \mathcal{F}(pos_j) - \hat{\text{GS}}_j \right\|_2. \tag{4}$$

Here, $\mathcal{P}$ represents the index set of Gaussians within a cluster, S denotes the importance score, $\mathcal{F}(\cdot)$ is the tiny MLP corresponding to the cluster, pos is the spatial position of the Gaussian, and $\hat{\text{GS}}$ is the normalized Gaussian attributes.

Difference between neural fields and quantization. Note that our neural field representations are more compact than simple quantization [10, 30]. Our approach employs multiple neural fields based on clustering, allowing predictions to vary with spatial positions, which means that we learn a compact function to map locations to different attributes. In contrast, vector quantization typically relies on a shared codebook to map similar attributes to the same index, limiting the flexibility of attribute fitting.

3.5. Fine-tuning

Directly using Gaussian attributes decoded from the neural field can result in significant degradation of rendering quality. To address this, we incorporate a fine-tuning stage to restore image quality. In this process, we fix the spatial positions of the 3D Gaussians and only fine-tune the tiny MLPs corresponding to each cluster. The photorealistic loss $\mathcal{L}_{\text{render}}$, is then computed by combining $\mathcal{L}_1$ and the SSIM loss $\mathcal{L}_{\text{SSIM}}$ with the weight λ as follows:

$$\mathcal{L}_{\text{render}} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{SSIM}}. \tag{5}$$

Frequency loss. We observe that during fine-tuning of tiny MLPs, high-frequency details, such as dense grass, tend to become over-smoothed or even lost. To address this and complete training within limited iterations, we introduce a frequency loss to better preserve high-frequency details. Specifically, we use a 2D discrete fourier transform to convert the rendered image I and the ground truth I_{gt} into respective frequency representations F and F_{gt}. $F(u, v)$ can be further expressed in terms of amplitude $\left|F(u, v)\right|$ and phase $\angle F(u, v)$, where (u, v) denotes the coordinates in the frequency spectrum. We then introduce a high-pass filter in the frequency domain to extract high-frequency information, denoted as $\hat{F}(u, v)$ and $\hat{F}_{gt}(u, v)$. We define $\Delta\left|\hat{F}(u, v)\right| = \left|\hat{F}(u, v)\right| - \left|\hat{F}_{gt}(u, v)\right|$ and $\Delta\angle\hat{F}(u, v) = \angle\hat{F}(u, v) - \angle\hat{F}_{gt}(u, v)$. Thus, the frequency loss $\mathcal{L}_{\text{freq}}$ and the total loss $\mathcal{L}_{\text{total}}$ can be formulated as follows:

$$\mathcal{L}_{\text{freq}} = \sum_{u=0}^{H-1}\sum_{v=0}^{W-1} \left|\Delta\left|\hat{F}(u, v)\right|\right| + \left|\Delta\angle\hat{F}(u, v)\right|, \tag{6}$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{render}} + \lambda_{\text{freq}}\mathcal{L}_{\text{freq}}. \tag{7}$$

Here, H, W and λ_{freq} denote the image height, width, and the hyperparameter to balance the loss.

Model parameters. In the end, we only need to store the positions of the pruned 3D Gaussians and the fine-tuned MLP weights for each cluster, significantly reducing the model size. Subsequently, the MLP can be utilized to decode the other attributes of the Gaussians for rendering.

4. Experiments

4.1. Experimental Settings

Evaluation Datasets and Metrics. We adopt four datasets for comparison. (1) *Mip-NeRF360* [3] offers scene-scale data for view synthesis, containing nine real-world large-scale scenes: five unbounded outdoor scenes and four indoor scenes with complex backgrounds. (2) *Tank and Temple* [17] is a unbounded dataset that includes two scenes: *train* and *truck*. (3) *Deep Blending* [13] contains two indoor scenes: *drjohnson* and *playroom*. (4) *NeRF Synthetic* [27] contains eight small-scale objects. For all datasets, we maintain the same train-test splits as the official setting of 3DGS [16] and utilize PSNR, SSIM [37], LPIPS [44], and model size to evaluate image quality and compression ratio.

Baselines. We use 3DGS [16] as our baseline method and compare with recent compression techniques [10, 11, 18, 28, 30]. For qualitative and quantitative comparisons, we use the official code provided for each method, along with their default configurations for training and rendering.

Implementation Details. We implement our NeuralGS based on the official codes of 3DGS [16] and conduct training on various scenes using NVIDIA A100 GPUs. During

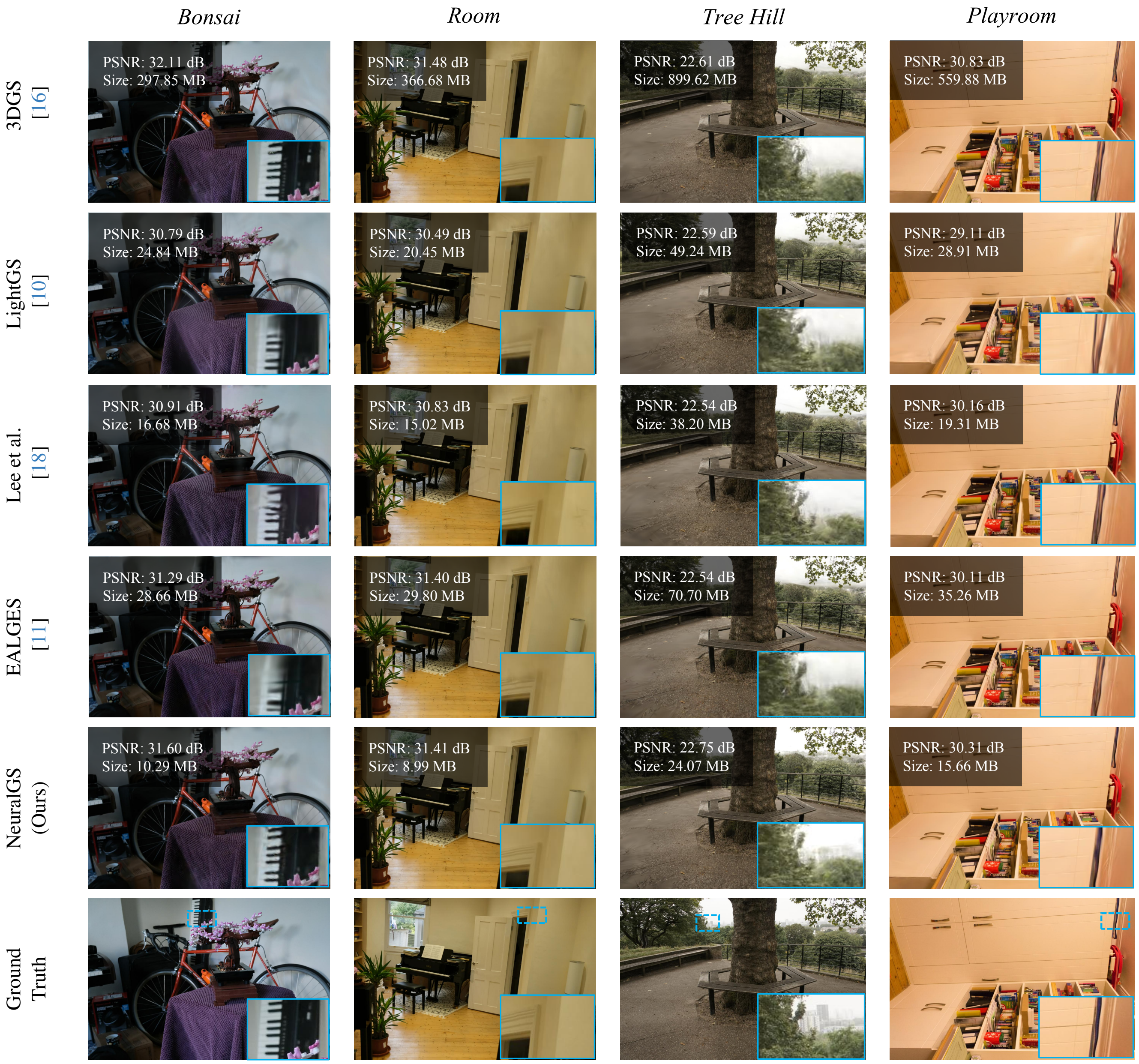


Figure 5. Qualitative results of the proposed method compared to 3DGS and existing compression methods.

pruning, we remove 40% of the less important 3D Gaussians to reduce the total number of Gaussians. For indoor scenes, we typically set the number of clusters K between 40 and 80, while for outdoor scenes, K is generally set between 100 and 140. Each cluster is assigned a tiny MLP to fit the Gaussian attributes for 60k iterations. All MLPs used in our method are 5-layer MLPs with Tanh activation function and positional encoding, and the hidden layer dimensions are uniformly set to 128. To restore rendering quality, we further fine-tuned the fitted MLPs for 25k iterations, with λ and λ_{freq} set to 0.2 and 0.01, respectively. Please refer to our supplementary materials for training results and specific implementation details.

4.2. Experimental Results

4.2.1. Quantitative Results.

The quantitative evaluation results across different datasets are presented in Tables 1 and Table 2. Specifically, compared to the original 3DGS [16], our method achieves significant compression ratios while preserving rendering quality. On the *Mip-NeRF 360* dataset and *Deep Blending* dataset, our method reduces the model size by ap-

Table 1. Quantitative results evaluated on Mip-NeRF 360 [3], Tanks&Temples [17], and Deep Blending [13] datasets. We highlight the best-performing results in red and the second-best results in yellow for all compression methods.

Dataset	Mip-NeRF 360 [3]				Tanks&Temples [17]				Deep Blending [13]			
Method	PSNR	SSIM	LPIPS	Storage	PSNR	SSIM	LPIPS	Storage	PSNR	SSIM	LPIPS	Storage
Mip-NeRF 360 [3] CVPR 2022	27.69	0.795	0.238	9.0 MB	22.16	0.757	0.261	9.0 MB	29.01	0.895	0.255	8.6 MB
3DGS [16] TOG 2023	27.48	0.812	0.222	755.5 MB	23.66	0.844	0.178	438.9 MB	29.42	0.900	0.247	672.8 MB
CompressGS [30] CVPR 2024	26.98	0.801	0.242	28.72 MB	23.32	0.830	0.194	17.73 MB	29.40	0.899	0.252	25.96 MB
Lee et al. [18] CVPR 2024	27.01	0.797	0.248	48.80 MB	23.29	0.829	0.202	39.43 MB	29.71	0.900	0.257	43.21 MB
CompactGS [28] ECCV 2024	25.95	0.780	0.267	30.43 MB	22.68	0.813	0.221	18.70 MB	28.90	0.891	0.282	14.28 MB
EAGLES [11] ECCV 2024	27.18	0.809	0.241	60.82 MB	23.27	0.835	0.211	31.05 MB	29.78	0.907	0.249	58.55 MB
LightGS [10] NeurIPS 2024	26.93	0.798	0.250	48.71 MB	22.92	0.817	0.242	24.74 MB	27.11	0.872	0.309	33.45 MB
NeuralGS (Ours)	27.21	0.798	0.249	19.54 MB	23.41	0.832	0.197	13.94 MB	29.84	0.906	0.254	17.31 MB

Table 2. Quantitative results of the proposed method evaluated on the NeRF-Synthetic [27] dataset. We highlight the best results in red and second-best results in yellow for compression methods.

Dataset	NeRF Synthetic Dataset [27]			
Method	PSNR	SSIM	LPIPS	Storage
Mip-NeRF 360 [3]	32.44	0.961	0.048	4.62 MB
3DGS [16]	33.75	0.970	0.031	69.89 MB
CompressGS [30]	32.94	0.967	0.033	3.82 MB
Lee et al. [18]	33.10	0.962	0.038	5.54 MB
CompactGS [28]	31.04	0.954	0.050	2.20 MB
EAGLES [11]	32.54	0.963	0.039	5.78 MB
LightGS [10]	32.70	0.963	0.039	7.84 MB
NeuralGS (Ours)	33.23	0.965	0.036	1.74 MB

proximately 39×, while on the *Tanks&Templates* dataset, it achieves about 32× model size reduction. Furthermore, our approach achieves the highest PSNR rendering metrics across all three datasets, outperforming existing compression methods [10, 11, 18, 28, 30] and even surpassing the original 3DGS by 0.42 dB on the *Deep Blending* dataset. These advancements are primarily attributed to the integration of 3DGS with cluster-based neural fields, which effectively facilitate compact representations of 3D scenes.

Table 2 presents the quantitative results on the *NeRF-Synthetic* dataset. Consistent with our previous observations, our method significantly reduces model storage from 69.9 MB to 1.7 MB, achieving an impressive 40× compression ratio while maintaining rendering quality comparable to the original 3DGS [16]. Moreover, compared to existing methods, our approach demonstrates substantial improvements in bitrate consumption. Detailed results of each scene are provided in the supplementary materials to further validate the advancements of our method.

4.2.2. Qualitative Results.

Figure 5 presents a qualitative comparison between our proposed NeuralGS and other Gaussian-based compression methods [10, 11, 18, 28, 30], providing the specific details with zoomed-in views. By leveraging compact cluster-based neural fields to encode the attributes of 3D Gaussians, our method greatly retains rendering quality with sharper

Table 3. Performance comparison with 3DGS [16]. Rendering FPS and model size (MB) are reported. The rendering speed of both methods is measured on our machine.

Method	Mip-NeRF 360		Tanks&Temples		Deep Blending	
	FPS	Size	FPS	Size	FPS	Size
3DGS [16]	112	756	162	439	118	673
NeuralGS	**135**	**19.5(39× ↓)**	**211**	**13.9(32× ↓)**	**137**	**17.3(39× ↓)**

textures and edges even using reduced model size.

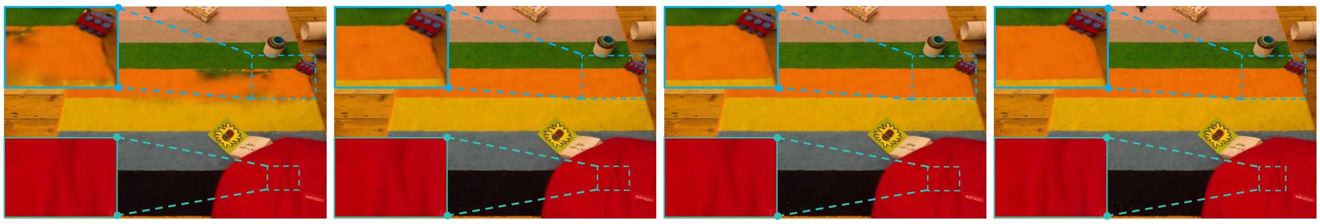
4.2.3. Rendering Time

As shown in Table 3, we compare the average storage size and rendering speed with the original 3DGS [16]. For rendering speed, we measure the frame rate or Frames Per Second (FPS) based on the total time taken to render all camera views in the dataset. Since we use multiple neural fields to encode Gaussian attributes, MLPs are used to decode the attributes of all 3D Gaussians before testing FPS, which constitutes a one-time amortized cost for loading the attributes. From Table 3, it is observed that, due to the reduced number of 3D Gaussians by pruning, our method achieves higher rendering speed compared to 3DGS while requiring significantly less model size with compact neural fields.

4.3. Ablation Studies

In this subsection, we conduct ablation studies on the *Deep Blending* dataset to demonstrate the effectiveness of each improvement. Specifically, our core idea is to use neural fields to encode Gaussian attributes, enhancing the compactness of 3D representation. Hence, our vanilla NeuralGS employs a single tiny MLP to fit the Gaussian attributes of the entire scene, followed by basic fine-tuning to restore quality. As shown in Table 4, we incrementally incorporate each improvement to validate the effectiveness of our approach. The ablation study on pruning will be presented in the appendix of the supplementary materials.

Effectiveness of Cluster-based Fitting. As shown in Table 4, the Vanilla NeuralGS results in significant degradation of 3D scene rendering quality compared to the original 3DGS [16]. This is primarily due to the large varia-



85% *90%* *95%* *100%*

Figure 6. NeuralGS allows progressive loading new clusters in the *playroom* scene to obtain more details and sharper texture.

w/o IW *w/* IW *w/o* FL *w/* FL

Figure 7. Ablation study about the impact of importance weight (IW) and frequency loss (FL) in the *bicycle* and *stump* scenes.

tions of 3D Gaussians, where a single tiny MLP tends to produce substantial fitting errors. To address this issue, we designed a clustering approach based on Gaussian attributes to maintain similarity within each cluster and assigned different tiny MLPs to fit the Gaussians of each cluster. As shown in Table 4, utilizing different tiny neural fields for clusters significantly reduces fitting errors, leading to 5 dB improvement in PSNR and 10% increase in SSIM, thereby substantially enhancing rendering quality.

Effectiveness of Importance weight. Notably, it is unnecessary to equally fit every Gaussian in the scene. Instead, we use the importance score of each Gaussian to represent its contribution to the renderings. This importance score is applied as a weight for the tiny MLP of each cluster during the fitting process, ensuring that important Gaussians are fitted by the neural fields with higher accuracy. As shown in Table 4 and Figure 7, adding importance scores as fitting weights, without introducing additional parameters, can further enhance quality and provide better textures.

Effectiveness of Frequency loss. During the fine-tuning stage, we observed that within a limited number of training iterations, MLPs tend to be less sensitive to high-frequency details. As shown in the second row of Figure 7, incorporating the frequency loss helps transform the blurry edges of leaves to be sharper. The quantitative results in Table 4 further demonstrate the improvement in rendering quality achieved by introducing the frequency loss.

Table 4. Quantitative ablation study on the Deep Blending [13] dataset by *progressively* adding our proposed improvement.

Dataset	Deep Blending Dataset [27]			
Method	PSNR	SSIM	LPIPS	Storage
3DGS [16]	29.42	0.900	0.247	672.8 MB
Vanilla NeuralGS	23.54	0.795	0.523	8.82 MB
+ Cluster-based fitting	28.82	0.891	0.294	17.31 MB
+ Importance weight	29.64	0.903	0.269	17.32 MB
+ Frequency loss (**Ours**)	**29.80**	**0.905**	**0.255**	**17.31 MB**

4.4. JPEG-like Progressive Loading

Benefiting from our use of different neural fields to fit the Gaussians within each cluster, we can transmit and decode Gaussian attributes cluster by cluster in a streamable manner like JPEG [34]. Specifically, we can sort clusters based on the number of Gaussians and progressively transmit the positions along with the corresponding tiny MLP weights. During transmission, Gaussian attributes can be decoded simultaneously, as shown in Figure. 6, enabling a progressive loading for the entire scene and making it suitable for streamable applications. From the magnified images, it is evident that newly loaded clusters contribute additional details, allowing the scene to gradually become clearer.

5. Future Work

Our current work primarily focuses on 3D scene reconstruction. Considering the rapid advancements in 4D scene reconstruction, future work could extend to 4D scenes, focusing on leveraging neural fields to further compress time-dependent 4D scenes and reduce the memory requirements.

6. Conclusion

In this paper, we introduce *NeuralGS*, a novel and effective post-compression approach for 3D Gaussian splatting. The core of our approach lies in leveraging compact neural fields to encode the attributes of 3D Gaussians with MLPs, significantly reducing the memory requirements of 3DGS. Thus, we design multiple neural fields based on clusters and incorporate importance scores as fitting weights to enhance the fittting quality of Gaussian attributes. Additionally, we introduce frequency loss during the fine-tuning stage to further preserve high-frequency details. Extensive experiments demonstrate that our method achieves comparable or even superior performance to existing compression methods while utilizing less model size. Overall, NeuralGS significantly alleviates the storage challenges of 3DGS, paving the way for its broader application in large-scale scenes.

References

[1] Muhammad Salman Ali, Sung-Ho Bae, and Enzo Tartaglione. Elmgs: Enhancing memory and computation scalability through compression for 3d gaussian splatting. *arXiv preprint arXiv:2410.23213*, 2024. 3

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *International Conference on Computer Vision (ICCV)*, 2021. 1

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 5, 7

[4] Junli Cao, Vidit Goel, Chaoyang Wang, Anil Kag, Ju Hu, Sergei Korolev, Chenfanfu Jiang, Sergey Tulyakov, and Jian Ren. Lightweight predictive 3d gaussian splats. *arXiv preprint arXiv:2406.19434*, 2024. 3

[5] 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*, 2024. 3

[6] Jianmei Dai, Zhilong Zhang, Shiwen Mao, and Danpu Liu. A view synthesis-based 360° vr caching system over mec-enabled c-ran. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(10):3843–3855, 2019. 1

[7] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4): 65–74, 1988. 1

[8] Engui Fan. Extended tanh-function method and its applications to nonlinear equations. *Physics Letters A*, 277(4-5): 212–218, 2000. 5

[9] Lue Fan, Yuxue Yang, Minxing Li, Hongsheng Li, and Zhaoxiang Zhang. Trim 3d gaussian splatting for accurate geometry representation. *arXiv preprint arXiv:2406.07499*, 2024. 3

[10] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. 2, 3, 4, 5, 6, 7

[11] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv preprint arXiv:2312.04564*, 2023. 2, 5, 6, 7

[12] Shrisudhan Govindarajan, Zeno Sambugaro, Towaki Takikawa, Daniel Rebain, Weiwei Sun, Nicola Conci, Kwang Moo Yi, Andrea Tagliasacchi, et al. Lagrangian hashing for compressed neural field representations. *arXiv preprint arXiv:2409.05334*, 2024. 2

[13] 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. 5, 7, 8

[14] 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. 2

[15] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19774–19783, 2023. 2

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 2023. 1, 2, 3, 4, 5, 6, 7, 8

[17] 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. 2, 5, 7

[18] 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. 2, 3, 5, 6, 7

[19] Jae Yong Lee, Yuqun Wu, Chuhang Zou, Derek Hoiem, and Shenlong Wang. Plenoptic png: Real-time neural radiance fields in 150 kb. *arXiv preprint arXiv:2409.15689*, 2024. 2

[20] Hao Li, Jinfa Huang, Peng Jin, Guoli Song, Qi Wu, and Jie Chen. Weakly-supervised 3d spatial reasoning for text-based visual question answering. *IEEE Transactions on Image Processing*, 32:3367–3382, 2023. 2

[21] Hao Li, Curise Jia, Peng Jin, Zesen Cheng, Kehan Li, Jialu Sui, Chang Liu, and Li Yuan. Freestyleret: Retrieving images from style-diversified queries. *arXiv preprint arXiv:2312.02428*, 2023. 1

[22] Zhihao Liang, Qi Zhang, Wenbo Hu, Ying Feng, Lei Zhu, and Kui Jia. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. *arXiv preprint arXiv:2403.11056*, 2024. 2

[23] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36 (2):451–461, 2003. 4

[24] Bangya Liu and Suman Banerjee. Swings: Sliding window gaussian splatting for volumetric video streaming with arbitrary length. *arXiv preprint arXiv:2409.07759*, 2024. 2

[25] 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. 3

[26] 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. 2

[27] 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 (ECCV)*, 2020. 1, 2, 5, 7, 8

[28] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. *arXiv preprint arXiv:2312.13299*, 2023. 2, 3, 5, 7

[29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics (TOG)*, 41(4):102:1–102:15, 2022. 2

[30] 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*, pages 10349–10358, 2024. 2, 3, 5, 7

[31] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[32] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 1

[33] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021. 2

[34] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001. 8

[35] 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. 2

[36] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 1

[37] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[38] Minye Wu and Tinne Tuytelaars. Implicit gaussian splatting with efficient multi-level tri-plane representation. *arXiv preprint arXiv:2408.10041*, 2024. 3

[39] 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. *arXiv preprint arXiv:2409.09756*, 2024. 3

[40] 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. 2

[41] Shenghai Yuan, Jinfa Huang, Yujun Shi, Yongqi Xu, Ruijie Zhu, Bin Lin, Xinhua Cheng, Li Yuan, and Jiebo Luo. Magictime: Time-lapse video generation models as metamorphic simulators. *arXiv preprint arXiv:2404.05014*, 2024. 2

[42] Shenghai Yuan, Jinfa Huang, Yongqi Xu, Yaoyang Liu, Shaofeng Zhang, Yujun Shi, Ruijie Zhu, Xinhua Cheng, Jiebo Luo, and Li Yuan. Chronomagic-bench: A benchmark for metamorphic evaluation of text-to-time-lapse video generation. *arXiv preprint arXiv:2406.18522*, 2024. 1

[43] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21424–21433, 2024. 2

[44] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[45] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 1

NeuralGS: Bridging Neural Fields and 3D Gaussian Splatting for Compact 3D Representations

Supplementary Material

7. Impletation Details

Having the supplementary compiled together with the main paper means that:

- The supplementary can back-reference sections of the main paper, for example, we can refer to Sec. 1;
- The main paper can forward reference sub-sections within the supplementary explicitly (e.g. referring to a particular experiment);
- When submitted to arXiv, the supplementary will already included at the end of the paper.

To split the supplementary pages from the main paper, you can use Preview (on macOS), Adobe Acrobat (on all OSs), as well as command line tools.