Compact 3D Gaussian Representation for Radiance Field

Joo Chan Lee[1] Daniel Rho[2] Xiangyu Sun[1] Jong Hwan Ko[1]✉ Eunbyung Park[1]✉
Sungkyunkwan University[1], KT[2]
[1]{maincold2@, xiangyusun@g., jhko@, epark@}skku.edu, [2]daniel.r@kt.com

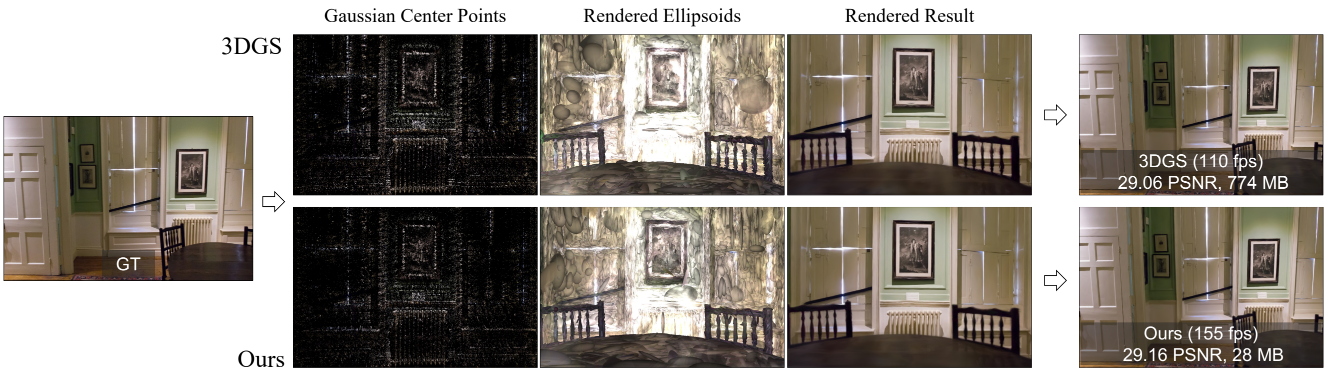


Figure 1. Our method achieves reduced storage and faster rendering speed while maintaining high-quality reconstruction of 3DGS [18]. The core idea is to effectively remove the redundant Gaussians that do not significantly contribute to the overall performance (the sparser distribution of Gaussian points and reduced ellipsoid redundancy shown in the figure). We also introduce a more compact representation of Gaussian attributes, resulting in markedly improved storage efficiency and rendering speed.

Abstract

Neural Radiance Fields (NeRFs) have demonstrated remarkable potential in capturing complex 3D scenes with high fidelity. However, one persistent challenge that hinders the widespread adoption of NeRFs is the computational bottleneck due to the volumetric rendering. On the other hand, 3D Gaussian splatting (3DGS) has recently emerged as an alternative representation that leverages a 3D Gaussisan-based representation and adopts the rasterization pipeline to render the images rather than volumetric rendering, achieving very fast rendering speed and promising image quality. However, a significant drawback arises as 3DGS entails a substantial number of 3D Gaussians to maintain the high fidelity of the rendered images, which requires a large amount of memory and storage. To address this critical issue, we place a specific emphasis on two key objectives: reducing the number of Gaussian points without sacrificing performance and compressing the Gaussian attributes, such as view-dependent color and covariance. To this end, we propose a learnable mask strategy that significantly reduces the number of Gaussians while preserving high performance. In addition, we propose a compact but effective representation of view-dependent color by employing a grid-based neural field rather than relying on spherical harmonics. Finally, we learn codebooks to compactly represent the geometric attributes of Gaussian by vector quantization. With model compression techniques such as quantization and entropy coding, we consistently show over 25× reduced storage and enhanced rendering speed, while maintaining the quality of the scene representation, compared to 3DGS. Our work provides a comprehensive framework for 3D scene representation, achieving high performance, fast training, compactness, and real-time rendering. Our project page is available at https://maincold2.github.io/c3dgs/.

✉ Corresponding authors.

1. Introduction

The field of neural rendering has witnessed substantial advancements in recent years, driven by the pursuit of rendering photorealistic 3D scenes from limited input data. Among the pioneering approaches, Neural Radiance Field (NeRF) [29] has gained considerable attention for its remarkable ability to generate high-fidelity images and 3D reconstructions of scenes from only a collection of 2D images in diverse applications. Follow-up research efforts have been dedicated to improving image quality [2, 31], accelerating training and rendering speed [7, 8, 10, 11, 31, 35], and reducing memory and storage footprints [23, 36, 41].

Despite the massive efforts, one persistent challenge that hinders the widespread adoption of NeRFs is the computational bottleneck due to the volumetric rendering. Since it demands dense point sampling along the ray to render a pixel, which requires significant computational resources, NeRFs often fail to achieve real-time rendering on handheld devices or low-end GPUs. This challenge limits their use in practical scenarios where fast rendering speed is essential, such as various interactive 3D applications.

3D Gaussian splatting (3DGS) [18] has recently emerged as an alternative representation that achieved very fast rendering speed and promising image quality. This approach leverages a point-based representation associated with 3D Gaussian attributes and adopts the rasterization pipeline to render the images rather than volumetric rendering. Highly optimized customized cuda kernels to maximize the parallelism and clever algorithmic tricks enable unprecedented rendering speed without compromising the image quality. However, a significant drawback arises as 3DGS entails a substantial number of 3D Gaussians to maintain the high-fidelity of the rendered images (Fig. 1), which requires a large amount of memory and storage (e.g., often $>$ 1GB for representing a large real-world scene).

To address the critical large memory and storage issue in 3DGS, we propose a compact 3D Gaussian representation framework. This approach significantly improves memory and storage efficiency while showing high-quality reconstruction, fast training speed, and real-time rendering, as shown in Fig. 1. We place a specific emphasis on two key objectives. First, we aim to reduce the number of Gaussian points required for scene representation without sacrificing performance. The densification process in 3DGS, which consists of cloning and splitting Gaussians, increases the number of Gaussians, and this is a crucial component in achieving a high level of detail. However, we observed that the current densification algorithm produces myriads of redundant and unnecessary Gaussians, resulting in high memory and storage requirements. We introduce a novel volume-based masking strategy that identifies and removes non-essential Gaussians that have minimal impact on overall performance. With the proposed masking method, we learn to reduce the number of Gaussians while achieving high performance during training. In addition to the efficient memory and storage usage, we can achieve faster rendering speed since the computational complexity is linearly proportional to the number of Gaussians.

Second, we propose compressing the Gaussian attributes, such as view-dependent color and covariance. In the original 3DGS, each Gaussian has its own attributes, and it does not exploit spatial redundancy, which has been widely utilized for various types of signal compression. For example, neighboring Gaussians may share similar color attributes, and we can reuse similar colors from neighboring Gaussians. Given this motivation, we incorporate a grid-based neural field to efficiently represent view-dependent colors rather than using per-Gaussian color attributes. When provided with the query Gaussian points, we extract the color attribute from the compact grid representation, avoiding the need to store it for each Gaussian separately. For our initial approach, we opt for a hash-based grid representation (Instant NGP [31]) from among several candidates due to its compactness and fast processing speed. This choice has led to a significant reduction in the spatial complexity of 3DGS.

In addition, 3DGS represents a scene with numerous small Gaussians collectively, and each Gaussian primitive is not expected to show high diversity. We found that the majority of Gaussians end up having similar geometry, with limited variation in scale and rotation. Based on this, we propose a codebook-based approach for modeling the geometry of Gaussians. It learns to find similar patterns or geometry shared across each scene and only stores the codebook index for each Gaussian, resulting in a very compact representation. Moreover, we found that a compact codebook suffices to represent a highly detailed scene, therefore, the spatial and computational overhead can be insignificant.

We have extensively tested our proposed compact 3D Gaussian representation on various datasets including real and synthetic scenes. Throughout the experiments regardless of dataset, our proposed method consistently showed about 15$\times$ reduced storage and enhanced rendering speed, while maintaining the quality of the scene representation, compared to 3DGS. Furthermore, our method can benefit from simple post-processing techniques such as quantization and entropy coding, consequently achieving over 25$\times$ compression throughout the datasets. Especially on a real-world dataset, Deep Blending [16], our method outperforms 3DGS in terms of reconstruction quality (measured in PSNR), notably with an over 28$\times$ enhancement in storage efficiency and a nearly 40% increase in rendering speed, setting a new state-of-the-art benchmark.

2. Related Work

2.1. Neural Radiance Fields

Neural radiance fields (NeRFs) have significantly expanded the horizons of 3D scene reconstruction and novel view synthesis. NeRF [29] introduced a novel approach to synthesizing novel views of 3D scenes, representing volume features by utilizing Multilayer Perceptrons (MLPs) and introducing volumetric rendering. Since its inception, various works have been proposed to enhance performance in diverse scenarios, such as different resolutions of reconstruction [2, 3], the reduced number of training samples [33, 37, 43, 45, 48], and reconstruction of large realistic scenes [30, 42] and dynamic scenes [12, 24, 25, 34]. However, NeRF's reliance on MLP has been a bottleneck, particularly causing slow training and inference.

In an effort to address the limitations, grid-based methods emerged as a promising alternative. These approaches using explicit voxel grid structures [4, 9, 10, 26, 27, 40] have demonstrated a significant improvement in training speed compared to traditional MLP-based NeRF methods. Nevertheless, despite this advancement, grid-based methods still suffer from relatively slow inference speeds and, more importantly, require large amounts of memory. This has been a substantial hurdle in advancing towards more practical and widely applicable solutions.

Subsequent research efforts have been directed toward the reduction of the memory footprint while maintaining or even enhancing the performance quality by grid factorization [6, 7, 11, 13, 15, 32], hash grids [31], grid quantization [39, 41] or pruning [10, 36]. These methods have also been instrumental in the fast training of 3D scene representation, thereby making more efficient use of computational resources. However, a persistent challenge that remains is the ability to achieve real-time rendering of large-scale scenes. The volumetric sampling inherent in these methods, despite their advancements, still poses a limitation.

2.2. Point-based Rendering and Radiance Field

Point-NeRF [46] employed points to represent a radiance field with volumetric rendering. Although it shows promising performance, the volume rendering hinders the real-time rendering. NeRF-style volumetric rendering and point-based α-blending fundamentally share the same model for rendering images but differ significantly in their rendering algorithms [18]. NeRFs offer a continuous feature representation of the entire volume as empty or occupied spaces, which necessitate costly volumetric sampling to render a pixel, leading to high computational demands. In contrast, points provide an unstructured, discrete representation of a volume geometry by the creation, destruction, and movement of points, and a pixel is rendered by blending several ordered points overlapping the pixel. This is achieved by optimizing opacity and positions [20], thus bypassing the limitations inherent in full volumetric representations and achieving remarkably fast rendering.

Point-based methods have been widely used in rendering 3D scenes, where the simplest form is point clouds. However, point clouds can lead to visual artifacts such as holes and aliasing. To mitigate this, point-based neural rendering methods have been proposed, processing the points through rasterization-based point splatting and differentiable rasterization [22, 44, 47]. The points were represented by neural features and rendered with CNNs [1, 20, 28]. However, these methods heavily rely on Multi-View Stereo (MVS) for initial geometry, inheriting its limitations, especially in challenging scenarios like areas lacking features, shiny surfaces, or fine structures.

Neural Point Catacaustics [21] addressed the issue of view-dependent effect through the use of an MLP, yet it still depends on MVS geometry for its input. Without the need for MVS, Zhang et al. [50] incorporated Spherical Harmonics (SH) for directional control. However, this method is constrained to managing scenes with only a single object and requires the use of masks during its initialization phase. Recently, 3D Gaussian Splatting (3DGS) [18] proposed using 3D Gaussians as primitives for real-time neural rendering. 3DGS utilizes highly optimized custom CUDA kernels and ingenious algorithmic approaches to achieve unparalleled rendering speed without sacrificing image quality.

While 3DGS does not require dense sampling for each ray, it does require a substantial number of 3D Gaussians to maintain a high level of quality in the resulting rendered images. Additionally, since each Gaussian consists of several rendering-related attributes like covariance matrices and SH with high degrees, 3DGS demands significant memory and storage resources, e.g., exceeding 1GB for a realistic scene. Our work aims to alleviate this parameter-intensive requirement while preserving high rendering quality, fast training, and real-time rendering.

3. Method

Background. In our approach, we build upon the foundation of 3D Gaussian Splatting (3DGS) [18], a point-based representation associated with 3D Gaussian attributes for representing 3D scenes. Each Gaussian represents 3D position, opacity, geometry (3D scale and 3D rotation represented as a quaternion), and spherical harmonics (SH) for view-dependent color. 3DGS constructs initial 3D Gaussians derived from the sparse data points obtained by Structure-from-Motion (SfM), such as COLMAP [38]. These Gaussians are cloned, split, pruned, and refined towards enhancing the anisotropic covariance for a precise depiction of the scene. This training process is based on the gradients from the differentiable rendering without unnecessary computation in empty areas, which accelerates training and rendering. However, 3DGS's high-quality reconstruction comes at the cost of memory and storage requirements, particularly with numerous Gaussians increased during training and their associated attributes.

Overall architecture. Our primary objectives are to 1) reduce the number of Gaussians and 2) represent attributes compactly while retaining the original performance. To this end, along with the optimization process, we mask out Gaussians that minimally impact performance, as shown in Fig. 2. For geometry attributes, such as scale and rotation, we propose using a codebook-based method that can fully exploit the limited variations of these attributes. We represent the color attributes using a grid-based neural field rather than storing their large parameters directly per each Gaussian. Finally, a small number of Gaussians with com-

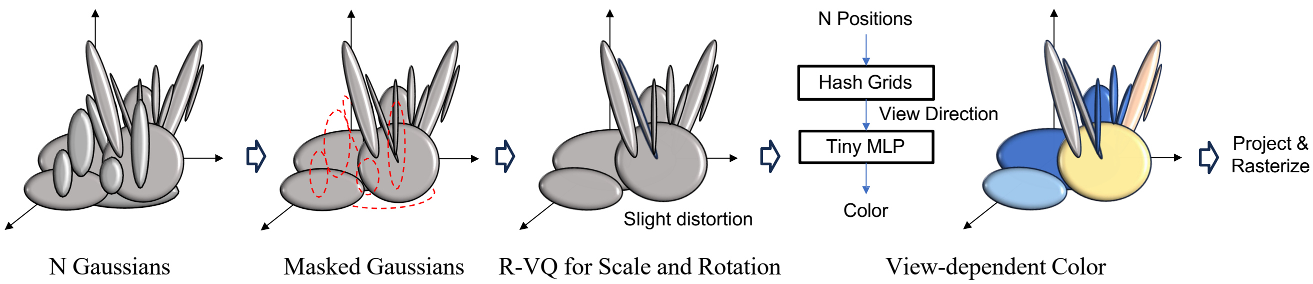


Figure 2. The detailed architecture of our proposed compact 3D Gaussian.

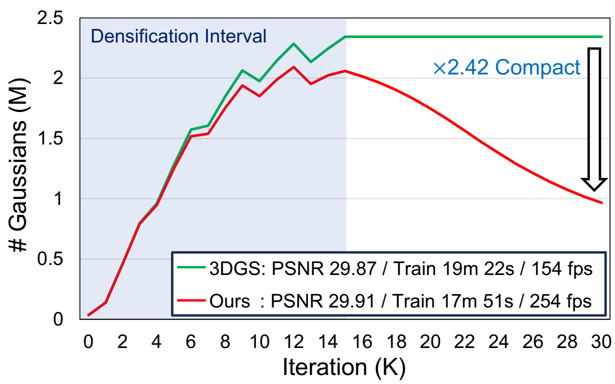


Figure 3. Visualization of the varying count of Gaussians during training (*Bonsai* scene). '# Gaussians' denotes the number of Gaussians.

pact attributes are then used for the subsequent rendering steps, including projection and rasterization to render images.

3.1. Gaussian Volume Mask

3DGS originally densifies Gaussians with large gradients by cloning or splitting. To regulate the increase in the number of Gaussians, opacities are set to a small number at every specific interval, and after some iterations, those with still minimal opacities are removed. Although this opacity-based control effectively eliminates some floaters, we empirically found that redundant Gaussians still exist significantly ($\times 2.42$ Gaussians show similar performance in Fig. 3).

The scale attribute of each Gaussian determines its 3D volume, which is then reflected in the rendering process. Small-sized Gaussians, due to their minimal volume, have a negligible contribution to the overall rendering quality, often to the point where their effect is essentially imperceptible. In such cases, it becomes highly beneficial to identify and remove such unessential Gaussians.

To this end, we propose a learnable masking of Gaussians based on their volume as well as transparency. We apply binary masks not only on the opacities $o \in [0,1]^N$ but also on the non-negative scale attributes $s \in \mathbb{R}_+^{N\times 3}$ that determine the volume geometry of N Gaussians, where N may vary with densification of Gaussians. We introduce an additional mask parameter $m \in \mathbb{R}^N$, based on which we generate binary masks $M \in \{0,1\}^N$. As it is not feasible to calculate gradients from binarized masks, we employ the straight-through estimator [5]. More specifically, the masked scale $\hat{s} \in \mathbb{R}_+^{N\times 3}$ and the masked opacity $\hat{o} \in [0,1]^N$ are formulated as follows,

$$M_n = \mathrm{sg}(\mathbb{1}[\sigma(m_n) > \epsilon] - \sigma(m_n)) + \sigma(m_n), \tag{1}$$

$$\hat{s}_n = M_n s_n, \quad \hat{o}_n = M_n o_n, \tag{2}$$

where n is the index of the Gaussian, ϵ is the masking threshold, $\mathrm{sg}(\cdot)$ is the stop gradient operator, and $\mathbb{1}[\cdot]$ and $\sigma(\cdot)$ are indicator and sigmoid function, respectively. This method allows for the incorporation of masking effects based on Gaussian volume and transparency in rendering. Considering both aspects together leads to more effective masking compared to considering either aspect alone.

We balance the accurate rendering and the number of Gaussians eliminated during training by adding masking loss L_m as follows,

$$L_m = \frac{1}{N}\sum_{n=1}^{N}\sigma(m_n). \tag{3}$$

At every densification step, we eliminate Gaussians based on the binary mask. Furthermore, unlike the original 3DGS, which stops densifying in the middle of the training and retains the number of Gaussians to the end, we consistently mask out throughout the entire training process, reducing unessential Gaussians effectively and ensuring efficient computation with low GPU memory throughout the training phase (Fig. 3). Once training is completed, we do not need to store the mask parameter m as we have already removed masked Gaussians.

3.2. Geometry Codebook

A number of Gaussians collectively construct a single scene, where similar geometric components can be shared throughout the entire volume. We have observed that the geometrical shapes of most Gaussians are very similar, showing only minor differences in scale and rotation. In addition, a scene is composed of many small Gaussians, and each Gaussian primitive is not expected to exhibit a wide

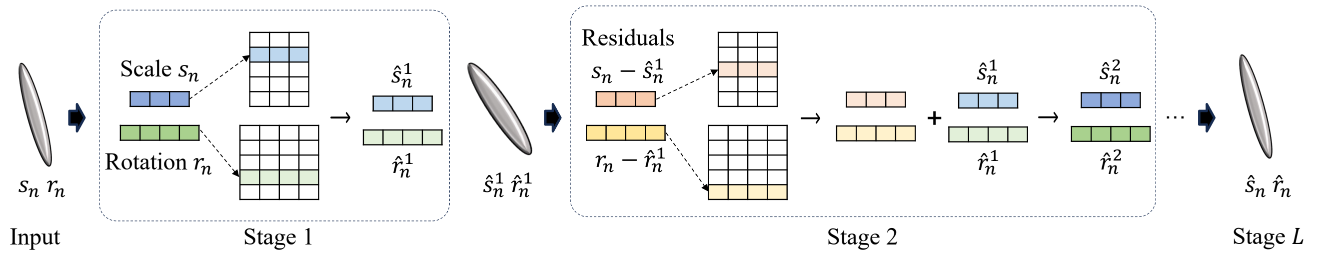


Figure 4. The detailed process of R-VQ to represent the scale and rotation of Gaussians. In the first stage, the scale and rotation vectors are compared to codes in each codebook, with the closest code identified as the result. In the next stage, the residual between the original vector and the first stage's result is compared with another codebook. This process is repeated up to the final stage, as a result, the selected indices and the codebook from each stage collectively represent the original vector.

range of diversity. Given this motivation, we propose a codebook learned to represent representative geometric attributes, including scale and rotation, by employing vector quantization (VQ) [14]. As naively applying vector quantization requires a high level of computational complexity and GPU memory [49], we adopt residual vector quantization (R-VQ) [49] that cascades L stages of VQ with codebook size C (Fig. 4), formulated as follows,

$$\hat{r}_n^l = \sum_{k=1}^{l} \mathcal{Z}^k[i^k], \quad l \in \{1, ..., L\}, \tag{4}$$

$$i_n^l = \underset{k}{\operatorname{argmin}} \, ||\mathcal{Z}^l[k] - (r_n - \hat{r}_n^{l-1})||_2^2, \quad \hat{r}_n^0 = \vec{0} \tag{5}$$

where $r \in \mathbb{R}^{N \times 4}$ is the input rotation vector, $\hat{r}^l \in \mathbb{R}^{N \times 4}$ is the output rotation vector after l quantization stages, and n is the index of the Gaussian. $\mathcal{Z}^l \in \mathbb{R}^{C \times 4}$ is the codebook at the stage l, $i^l \in \{0, \dots, C-1\}^N$ is the selected indices of the codebook at the stage l, and $\mathcal{Z}[i] \in \mathbb{R}^4$ represents the vector at index i of the codebook $\mathcal{Z}$.

The objective function for training the codebooks is as follows,

$$L_r = \frac{1}{NC} \sum_{k=1}^{L} \sum_{n=1}^{N} ||\operatorname{sg}[r_n - \hat{r}_n^{k-1}] - \mathcal{Z}^k[i_n^k]||_2^2, \tag{6}$$

where $\operatorname{sg}[\cdot]$ is the stop-gradient operator. We use the output from the final stage $\hat{r}^L$ (we will omit the superscript L for brevity from now onwards), and the R-VQ process is similarly applied to scale s before masking (we also similarly use the objective function for scale L_s).

3.3. Compact View-dependent Color

Each Gaussian in 3DGS requires 48 of the total 59 parameters to represent SH (max 3 degrees) to model the different colors according to the viewing direction. Instead of using the naive and parameter-inefficient approach, we propose representing the view-dependent color of each Gaussian by exploiting a grid-based neural field. To this end, we contract the unbounded positions $p \in \mathbb{R}^{N \times 3}$ to the bounded range, motivated by mip-NeRF 360 [3], and compute the 3D view direction $d \in \mathbb{R}^3$ for each Gaussian based on the camera center point. We exploit hash grids [31] followed by a tiny MLP to represent color. Here, we input positions into the hash grids, and then the resulting feature and the view direction are fed into the MLP. More formally, view-dependent color $c_n(\cdot)$ of Gaussian at position $p_n \in \mathbb{R}^3$ can be expressed as,

$$c_n(d) = f(\operatorname{contract}(p_n), d; \theta), \tag{7}$$

$$\operatorname{contract}(p_n) = \begin{cases} p_n & ||p_n|| \le 1 \\ \left(2 - \frac{1}{||p_n||}\right)\left(\frac{p_n}{||p_n||}\right) & ||p_n|| > 1, \end{cases} \tag{8}$$

where $f(\cdot;\theta), \operatorname{contract}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^3$ stand for the neural field with parameters θ, and the contraction function, respectively. We represent the 0-degree components of SH (the same number of channels as RGB, but not view-dependent) and then convert them into RGB colors due to the slightly increased performance compared to representing the RGB color directly.

3.4. Training

Here, we have N Gaussians and their attributes, position p_n, opacity o_n, rotation $\hat{r}_n$, scale $\hat{s}_n$, and view-dependent color $c_n(\cdot)$, which are used to render images. The entire model is trained end-to-end based on the rendering loss L_{ren}, the weighted sum of the L1 and SSIM loss between the GT and rendered images. By adding the loss for masking L_m and geometry codebooks L_r, L_s, the overall loss L is as follows,

$$L = L_{ren} + \lambda_m L_m + L_r + L_s, \tag{9}$$

where λ_m is a hyperparameter to regularize the number of Gaussians. To avoid heavy computational costs and ensure fast and optimal training, we initialize the learnable codebooks with K-means and apply R-VQ only during the last 1K training iterations. Except for that period, we set L_r, L_s to zero.

Table 1. Qualitative results of the proposed method evaluated on Mip-NeRF 360 and Tanks&Temples datasets. We reported the numbers of baselines from the original paper (denoted as 3DGS), which were run on an NVIDIA A6000 GPU. For a fair comparison, we re-evaluate 3DGS with the same training configurations as our method using an NVIDIA A100 GPU (denoted as 3DGS*).

Dataset	Mip-NeRF 360						Tanks&Temples					
Method	PSNR	SSIM	LPIPS	Train	FPS	Storage	PSNR	SSIM	LPIPS	Train	FPS	Storage
Plenoxels	23.08	0.626	0.463	25m 49s	6.79	2.1 GB	21.08	0.719	0.379	25m 05s	13.0	2.3 GB
INGP-base	25.30	0.671	0.371	05m 37s	11.7	13 MB	21.72	0.723	0.330	05m 26s	17.1	13 MB
INGP-big	25.59	0.699	0.331	07m 30s	9.43	48 MB	21.92	0.745	0.305	06m 59s	14.4	48 MB
Mip-NeRF 360	27.69	0.792	0.237	48h	0.06	8.6 MB	22.22	0.759	0.257	48h	0.14	8.6 MB
3DGS	27.21	0.815	0.214	41m 33s	134	734 MB	23.14	0.841	0.183	26m 54s	154	411 MB
3DGS*	27.46	0.812	0.222	24m 07s	120	746 MB	23.71	0.845	0.178	13m 51s	160	432 MB
Ours	27.08	0.798	0.247	33m 06s	**128**	**48.8 MB**	23.32	0.831	0.201	18m 20s	**185**	**39.4 MB**
Ours+PP	27.03	0.797	0.247	-	-	**29.1 MB**	23.32	0.831	0.202	-	-	**20.9 MB**

Table 2. Qualitative results of the proposed method evaluated on Deep Blending dataset. We re-evaluate 3DGS* same with our method.

Dataset	Deep Blending					
Method	PSNR	SSIM	LPIPS	Train	FPS	Storage
Plenoxels	23.06	0.795	0.510	27m 49s	11.2	2.7 GB
INGP-base	23.62	0.797	0.423	06m 31s	3.26	13 MB
INGP-big	24.96	0.817	0.390	08m 00s	2.79	48 MB
Mip-NeRF 360	29.40	0.901	0.245	48h	0.09	8.6 MB
3DGS	29.41	0.903	0.243	36m 02s	137	676 MB
3DGS*	29.46	0.900	0.247	21m 52s	132	663 MB
Ours	**29.79**	**0.901**	0.258	27m 33s	**181**	**43.2MB**
Ours+PP	29.73	0.900	0.258	-	-	**23.8MB**

Table 3. Qualitative results of the proposed method evaluated on NeRF-Synthetic dataset. * denotes the reported value in the original paper.

Dataset	NeRF-Synthetic			
Method	PSNR	Storage	Train	FPS
3DGS	33.32*	68.1 MB	6m 14s	359
Ours	33.33	**5.55 MB** ($\times$0.08)	8m 04s ($\times$1.29)	**545** ($\times$1.52)
Ours+PP	32.88	**2.67 MB** ($\times$0.04)	-	-

4. Experiment

4.1. Implementation Details

We tested our approach on three real-world datasets (Mip-NeRF 360 [3], Tanks&Temples [19], and Deep Blending [16]) and a synthetic dataset (NeRF-Synthetic [29]). Following 3DGS, we chose two scenes from Tanks&Temples and Deep Blending. The models with the proposed method (denoted as Ours) were trained during 30K iterations, where we stored positions p and opacities o with 16-bit precision using half-tensors. Additionally, we implemented straightforward post-processing techniques on the model attributes, a variant we denote as Ours+PP. These post-processing steps include:

- Applying 8-bit min-max quantization to opacity and hash grid parameters.
- Pruning hash grid parameters with values below 0.1.
- Applying Huffman encoding [17] on the quantized opacity and hash parameters, and R-VQ indices.

Further implementation details are provided in the supplementary materials.

4.2. Performance Evaluation

Real-world scenes. Tab. 1 and Tab. 2 show the qualitative results evaluated on real-world scenes. Across datasets, our approach achieves high reconstruction performance comparable to 3DGS while drastically reducing the overall size and accelerating rendering. Especially for the Deep Blending dataset (Tab. 2), our method even outperforms the original 3DGS in terms of visual quality (measured in PSNR and SSIM), achieving state-of-the-art performance with the fastest rendering speed as well as compactness (almost 40% faster rendering and over 15$\times$ compactness compared to 3DGS). Comparing our method to 3DGS, the qualitative results in Fig. 5 further demonstrate our method's high reconstruction quality with substantially less size.

Synthetic scenes. We also evaluate our method on synthetic scenes. As 3DGS has proven its effectiveness in improving visual quality, rendering speed, and training time compared to other baselines, we focus on comparing our method with 3DGS, highlighting the improvements from our method. As shown in Tab. 3, although our approach requires slightly more training duration, we achieve over 10$\times$ compression and 50% faster rendering compared to 3DGS, maintaining high-quality reconstruction.

With post-processings, our model can be further downsized by over 40 % regardless of the dataset. Consequently, we achieve more than 25$\times$ compression from 3DGS, while maintaining high performance.

4.3. Ablation Study

Learnable volume masking. As shown in Tab. 4, the proposed volume-based masking significantly reduces the number of Gaussians while retaining (even slightly increas-

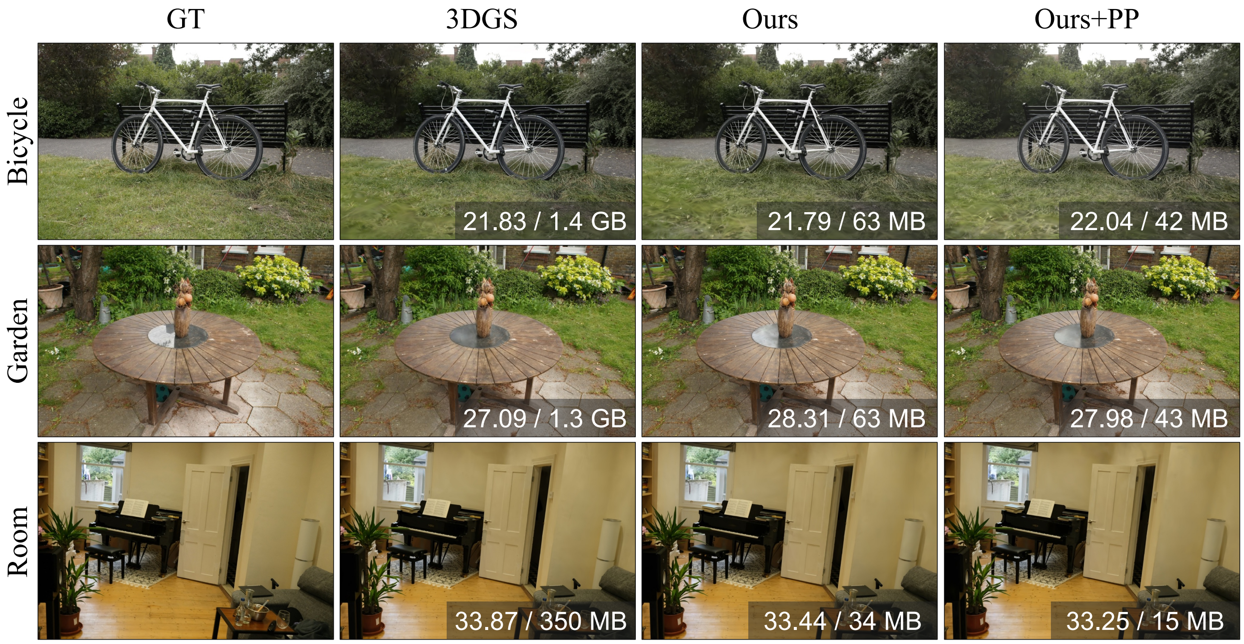


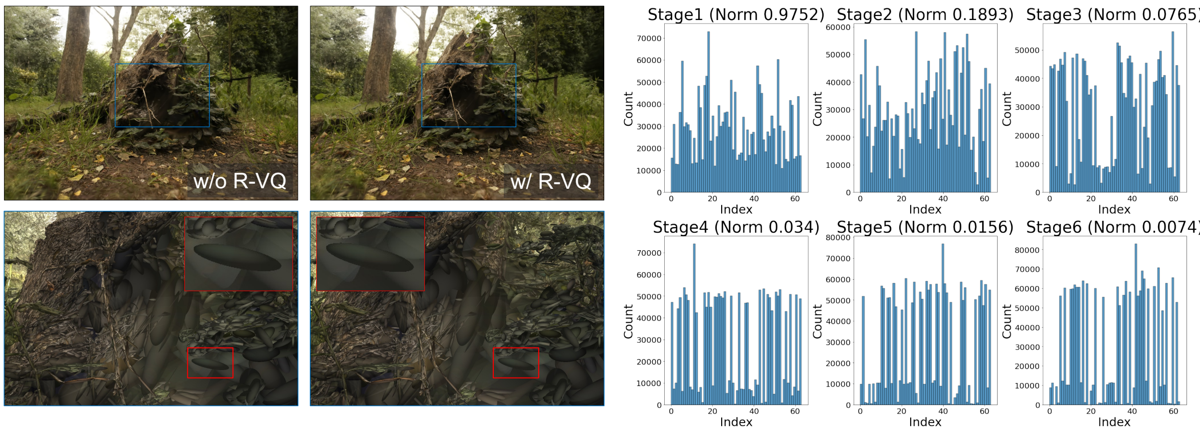
Figure 5. Qualitative results of our approach compared to 3DGS. We present the rendering PSNR and storage on the results.

Figure 6. Effect of the proposed learnable volume masking, compared to the original 3DGS. We visualize Gaussian center points, ellipsoids, and rendered results using the *Playroom* scene.

ing) the visual quality, demonstrating its effectiveness in removing redundant Gaussians. The reduced Gaussians show several additional advantages: reducing training time, storage, and testing time. Specifically on the *Playroom* scene, our proposed masking method shows a 140% increase in storage efficiency and a 65% increase in rendering speed. Fig. 6 further illustrates the effect of the masking method on Gaussians and the resulting images. Despite the noticeable reduction in the number of Gaussians, as evidenced by the sparser points in the visualization, the quality of the rendered results remains high, with no visible differences. These results demonstrate the effectiveness and efficiency of the proposed method, both quantitatively and qualitatively.

Compact view-dependent color. The proposed color representation based on the neural field offers more than a threefold improvement in storage efficiency with a slightly reduced number of Gaussians compared to directly storing high-degree SH, despite necessitating slightly more time for training and rendering. Nonetheless, when compared to 3DGS, the proposed color representation with the masking strategy demonstrates either a faster or comparable rendering speed.

Geometry codebook. Our proposed geometry codebook approach achieves a reduction in storage requirements by approximately 30% while maintaining the reconstruction quality, training time, and rendering speed. To understand the factors behind this enhanced performance, we begin with an analysis of the Gaussians visualized in Fig. 7-(a). As the figure shows, we can observe that the majority of Gaussians maintain their scales and rotations regardless of R-VQ, with only minor differences in a few that are hardly



(a) Rendered results

(b) Learned codebook indices for rotation

Figure 7. Effect of the proposed geometry codebook. We visualize (a) ellipsoids and rendered results, and (b) learned codebook indices for rotation using the *Stump* scene. 'Norm' denotes the average norm of all code vectors in each codebook (representing magnitude).

Table 4. Ablation study on the proposed contributions. 'M', 'C', 'G', and 'H' denote masking, color representation, geometry codebook, and half tensor for positions and opacities, respectively. '#Gauss' means the number of Gaussians.

Method \Dataset				Playroom					Bonsai				
M	C	G	H	PSNR	Train time	#Gauss	Storage	FPS	PSNR	Train time	#Gauss	Storage	FPS
3DGS				29.87	19m 22s	2.34 M	553 MB	154	32.16	19m 18s	1.25 M	295 MB	200
✓				29.91	17m 51s	967 K	228 MB	254	32.22	18m 50s	643 K	152 MB	247
✓	✓			30.33	23m 56s	770 K	59 MB	210	32.08	23m 09s	592 K	51 MB	196
✓	✓	✓		30.33	24m 58s	761 K	44 MB	204	32.08	24m 06s	598 K	40 MB	198
✓	✓	✓	✓	30.32	24m 35s	778 K	38 MB	206	32.08	24m 16s	601 K	35 MB	196

Table 5. Avarge storage (MB) for each Gaussian attribute, evaluated on Mip-NeRF 360 dataset. f, 8b, H, and P mean floating-point, min-max quantization to 8-bit, Huffman encoding, and pruning parameters below 0.1, respectively.

	Pos.	Opa.	Sca.	Rot.	Col.		Tot.
3DGS	32f						746
	37.9	12.6	37.9	50.6	606.9		
Ours	16f		R-VQ		Hash(16f)	MLP(16f)	48.8
	8.3	2.8	6.3	6.3	25.2	0.016	
Ours +PP	16f	8b+H	+H		+8b+P+H	MLP(16f)	29.1
	8.3	1.2	5.9	6.2	7.4	0.016	

noticeable. Beyond qualitative analysis, we also explore the patterns of learned indices across each stage of R-VQ, shown in Fig. 7-(b). The lower stages exhibit relatively even distributions with large magnitudes of codes. As the stages progress, the distribution gets uneven and the magnitude of codes decreases, indicating the residuals of each stage have been reduced and trained to represent geometry.

Effect of post-processing. Tab. 5 describes the size of each attribute with and without the application of post-processing techniques. While our end-to-end trainable framework demonstrates significant effectiveness, it requires relatively large storage for color representation. Nonetheless, as indicated in the table, this can be effectively reduced through simple post-processing.

5. Conclusion

We have proposed a compact 3D Gaussian representation for 3D scenes, reducing the number of Gaussians without sacrificing visual quality through a novel volume-based masking. Furthermore, this work proposed combining the neural field and exploiting the learnable codebooks to represent Gaussian attributes compactly. Through extensive experiments, our approach demonstrated more than a tenfold reduction in storage and a marked increase in rendering speed while retaining the high-quality reconstruction compared to 3DGS. This result sets a new benchmark with high visual quality, compactness, fast training, and real-time rendering. Our framework thus stands as a comprehensive solution, paving the way for broader adoption and application in various fields requiring efficient and high-quality 3D scene representation.

Acknowledgements

This work was supported by the Ministry of Science and ICT (MSIT) of Korea under the National Research Foundation (NRF) grants (RS-2023-00245342) and Institute of Information and Communication Technology Planning Evaluation (IITP) grants (IITP-2019-0-00421, IITP-2023-2020-0-01821, IITP-2021-0-02068, IITP-2021-0-02052), and by the Technology Innovation Program (RS-2023-00235718) funded by the Ministry of Trade, Industry & Energy (1415187474).

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *European Conference on Computer Vision*, pages 696–712, 2020. 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 1, 2

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 5, 6

[4] 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. 3

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

[6] 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. 3

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, 2022. 1, 3

[8] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 1

[9] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 3

[10] 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. 1, 3

[11] 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. 1, 3

[12] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 2

[13] 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. 3

[14] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1 (2):4–29, 1984. 5

[15] 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. 3

[16] 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, 6

[17] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9): 1098–1101, 1952. 6

[18] 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. 1, 2, 3

[19] 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. 6

[20] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, pages 29–43, 2021. 3

[21] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics (TOG)*, 41(6):1–15, 2022. 3

[22] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021. 3

[23] Joo Chan Lee, Daniel Rho, Seungtae Nam, Jong Hwan Ko, and Eunbyung Park. Coordinate-aware modulation for neural fields. In *International Conference on Learning Representations*, 2024. 1

[24] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 2

[25] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2

[26] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie,

and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *Advances in Neural Information Processing Systems*, 35:36762–36775, 2022. 3

[27] 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. 3

[28] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 3

[29] 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. 1, 2, 6

[30] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022. 2

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), 2022. 1, 2, 3, 5

[32] Seungtae Nam, Daniel Rho, Jong Hwan Ko, and Eunbyung Park. Mip-grid: Anti-aliased grid representations for neural radiance fields. In *Advances in Neural Information Processing Systems*, pages 2837–2849, 2023. 3

[33] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022. 2

[34] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[35] 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. 1

[36] 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. 1, 3

[37] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022. 2

[38] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[39] Seungjoo Shin and Jaesik Park. Binary radiance fields. In *Advances in Neural Information Processing Systems*, pages 55919–55931, 2023. 3

[40] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 3

[41] 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. 1, 3

[42] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 2

[43] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 2

[44] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 3

[45] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. Sinnerf: Training neural radiance fields on complex scenes from a single image. In *European Conference on Computer Vision*, pages 736–753, 2022. 2

[46] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 3

[47] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. 3

[48] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2

[49] 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. 5

[50] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 3