Spec-Gaussian: Anisotropic View-Dependent Appearance for 3D Gaussian Splatting

ZIYI YANG, Zhejiang University & ByteDance Inc., China
XINYU GAO, Zhejiang University, China
YANGTIAN SUN, The University of Hong Kong, Hong Kong
YIHUA HUANG, The University of Hong Kong, Hong Kong
XIAOYANG LYU, The University of Hong Kong, Hong Kong
WEN ZHOU, ByteDance Inc., China
SHAOHUI JIAO, ByteDance Inc., China
XIAOJUAN QI, The University of Hong Kong, Hong Kong
XIAOGANG JIN, Zhejiang University, China

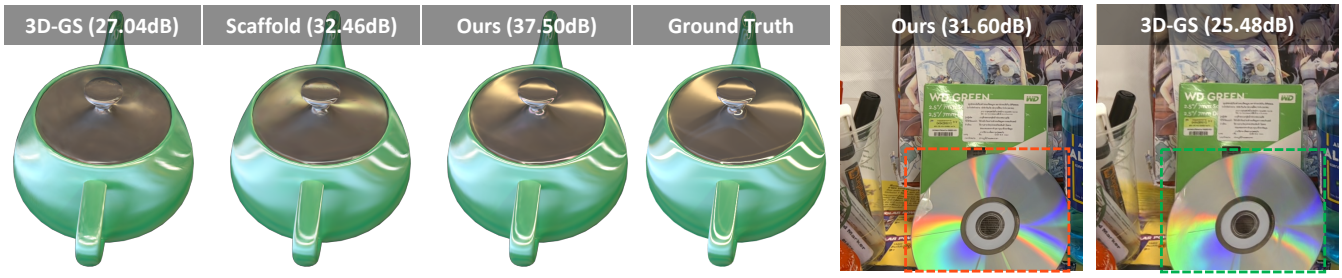

Fig. 1. Our method not only achieves real-time rendering but also significantly enhances the capability of 3D-GS to model scenes with specular and anisotropic components. Key to this enhanced performance is our use of ASG appearance field to model the appearance of each 3D Gaussian, which results in substantial improvements in rendering quality for both complex and general scenes. Moreover, we employ anchor Gaussians to constrain the geometry of point-based representations, thereby improving the ability of 3D-GS to accurately model reflective parts and accelerating both training and rendering processes.

The recent advancements in 3D Gaussian splatting (3D-GS) have not only facilitated real-time rendering through modern GPU rasterization pipelines but have also attained state-of-the-art rendering quality. Nevertheless, despite its exceptional rendering quality and performance on standard datasets, 3D-GS frequently encounters difficulties in accurately modeling specular and anisotropic components. This issue stems from the limited ability of spherical harmonics (SH) to represent high-frequency information. To overcome this challenge, we introduce *Spec-Gaussian*, an approach that utilizes an anisotropic spherical Gaussian (ASG) appearance field instead of SH for modeling the view-dependent appearance of each 3D Gaussian. Additionally, we have developed a coarse-to-fine training strategy to improve learning efficiency and eliminate floaters caused by overfitting in real-world scenes. Our experimental results demonstrate that our method surpasses existing approaches in terms of rendering quality. Thanks to ASG, we have significantly improved the ability of 3D-GS to model scenes with specular and anisotropic components without increasing the number of 3D Gaussians. This improvement extends the applicability of 3D GS to handle intricate scenarios with specular and anisotropic surfaces. Our codes and datasets will be released.

Additional Key Words and Phrases: 3D Gaussian Splatting, Anisotropic Spherical Gaussian, Real-time Rendering

ACM Reference Format:
Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. 2024. Spec-Gaussian: Anisotropic View-Dependent Appearance for 3D Gaussian Splatting. In . ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

High-quality reconstruction and photorealistic rendering from a collection of images are crucial for a variety of applications, such as augmented reality/virtual reality (AR/VR), 3D content production, and art creation. Classic methods employ primitive representations, like meshes [34] and points [4, 60], and take advantage of the rasterization pipeline optimized for contemporary GPUs to achieve real-time rendering. In contrast, neural radiance fields (NeRF) [5, 32, 33] utilize neural implicit representation to offer a continuous scene representation and employ volumetric rendering to produce rendering results. This approach allows for enhanced preservation of scene details and more effective reconstruction of scene geometries.

Recently, 3D Gaussian Splatting (3D-GS) [21] has emerged as a leading technique, delivering state-of-the-art quality and real-time speed. This method optimizes a set of 3D Gaussians that capture the appearance and geometry of a 3D scene simultaneously, offering a continuous representation that preserves details and produces high-quality results. Besides, the CUDA-customized differentiable rasterization pipeline for 3D Gaussians enables real-time rendering even at high resolution.

Despite its exceptional performance, 3D-GS struggles to model specular components within scenes (see Fig. 1). This issue primarily stems from the limited ability of low-order spherical harmonics (SH) to capture the high-frequency information required in these scenarios. Consequently, this poses a challenge for 3D-GS to model scenes with reflections and specular components, as illustrated in Fig. 1 and Fig. 7.

To address the issue, we introduce a novel approach called *Spec-Gaussian*, which combines anisotropic spherical Gaussian (ASG) [54] for modeling anisotropic and specular components, anchor-based geometry-aware 3D Gaussians for acceleration and storage reduction, and an effective training mechanism to eliminate floaters and improve learning efficiencies. Specifically, the method incorporates three key designs: 1) A new 3D Gaussian representation that utilizes an ASG appearance field instead of SH to model the appearance of each 3D Gaussian. ASG with a few orders can effectively model high-frequency information that low-order SH cannot. This new design enables 3D-GS to more effectively model anisotropic and specular components in static scenes. 2) A hybrid approach employing sparse anchor points to control the location and representation of its child Gaussians. This strategy results in a hierarchical and geometry-aware point-based scene representation and enables us to store only the anchor Gaussians, significantly reducing storage requirements and enhancing the geometry. 3) A coarse-to-fine training scheme specifically tailored for 3D-GS is designed to eliminate floaters and boost learning efficiency. This strategy effectively shortens learning time by optimizing low-resolution rendering in the initial stage, preventing the need to increase the number of 3D Gaussians and regularizing the learning process to avoid the generation of unnecessary geometric structures that lead to floaters.

By combining these advances, our approach can render high-quality results for specular highlights and anisotropy as shown in Fig. 4 while preserving the efficiency of Gaussians. Furthermore, comprehensive experiments reveal that our method not only endows 3D-GS with the ability to model specular highlights but also achieves state-of-the-art results in general benchmarks.

In summary, the major contributions of our work are as follows:

- A novel ASG appearance field to model the view-dependent appearance of each 3D Gaussian, which enables 3D-GS to effectively represent scenes with specular and anisotropic components without sacrificing rendering speed.
- An anchor-based hybrid model to reduce the computational and storage overhead brought by learning the ASG appearance field.
- A coarse-to-fine training scheme that effectively regularizes training to eliminate floaters and improve the learning efficiency of 3D-GS in real-world scenes.
- An anisotropic dataset has been made to assess the capability of our model in representing anisotropy. Extensive experiments show the effectiveness of our method in modeling scenes with specular highlights and anisotropy.

2 RELATED WORK

2.1 Implicit Neural Radiance Fields

Neural rendering has attracted significant interest in the academic community for its unparalleled ability to generate photorealistic images. Methods like NeRF [32] utilize Multi-Layer Perceptrons (MLPs) to model the geometry and radiance fields of a scene. Leveraging the volumetric rendering equation and the inherent continuity and smoothness of MLPs, NeRF achieves high-quality scene reconstruction from a set of posed images, establishing itself as the state-of-the-art (SOTA) method for novel view synthesis. Subsequent research has extended the utility of NeRF to various applications, including mesh reconstruction [25, 46, 52], inverse rendering [29, 42, 56, 63], optimization of camera parameters [27, 36, 47, 48], few-shot learning [12, 51, 55], and anti-aliasing [1–3].

However, this stream of methods relies on ray casting rather than rasterization to determine the color of each pixel. Consequently, every sampling point along the ray necessitates querying the MLPs, leading to significantly slow rendering speed and prolonged training convergence. This limitation substantially impedes their application in large-scene modeling and real-time rendering.

To reduce the training time of MLP-based NeRF methods and improve rendering speed, subsequent work has enhanced NeRF's efficiency in various ways. Structure-based techniques [8, 13, 16, 38, 61] have sought to improve inference or training efficiency by caching or distilling the implicit neural representation into more efficient data structures. Hybrid methods [28, 43] increase efficiency by incorporating explicit voxel-based data structures. Factorization methods [6, 9, 15, 17] apply a low-rank tensor assumption to decompose the scene into low-dimensional planes or vectors, achieving better geometric consistency. Compared to continuous implicit representations, the convergence of individual voxels in the grid is independent, significantly reducing training time. Additionally, Instant-NGP [33] utilizes a hash grid with a corresponding CUDA implementation for faster feature querying, enabling rapid training and interactive rendering of neural radiance fields.

Despite achieving higher quality and faster rendering, these methods have not fundamentally overcome the substantial query overhead associated with ray casting. As a result, a notable gap remains before achieving real-time rendering. In this work, we build upon the recent 3D-GS [21], a point-based rendering method that leverages rasterization. Compared to ray casting-based methods, it significantly enhances both training and rendering speed.

2.2 Point-based Neural Radiance Fields

Point-based representations, similar to triangle mesh-based methods, can exploit the highly efficient rasterization pipeline of modern GPUs to achieve real-time rendering. Although these methods offer breakneck rendering speeds and are well-suited for editing tasks, they often suffer from holes and outliers, leading to artifacts in the rendered images. This issue arises from the discrete nature of point clouds, which can create gaps in the primitives and, consequently, in the rendered image.

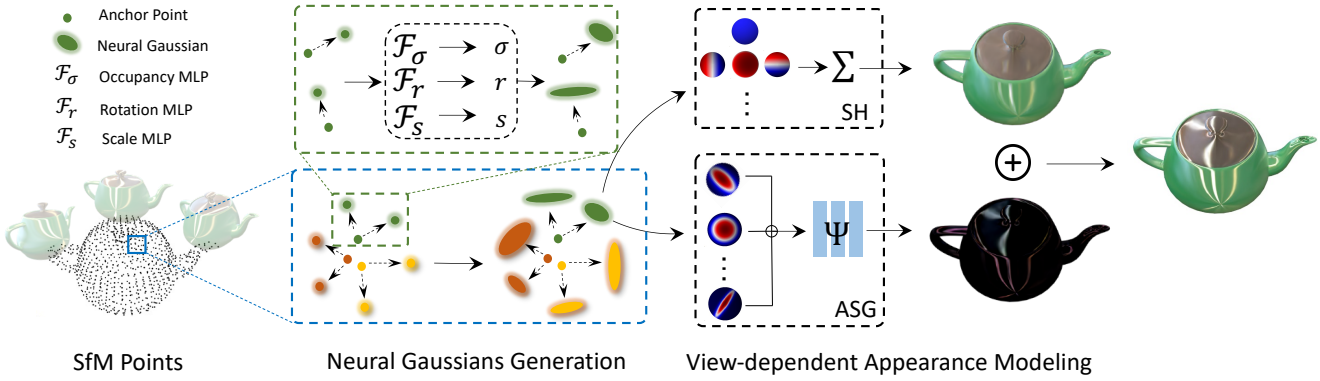

Fig. 2. **Pipeline of our proposed Spec-Gaussian.** The optimization process begins with SfM points derived from COLMAP or generated randomly, serving as the initial state for the anchor Gaussians. Within a view frustum, k neural Gaussians are spawned from each visible anchor Gaussian, using the corresponding offsets. Their other attributes, such as opacity σ, rotation r, and scaling s, are decoded through the respective tiny MLPs. To address the limitations of low-order SH and pure MLP in modeling high-frequency information, we additionally employ ASG in conjunction with a feature decoupling MLP to model the view-dependent appearance of each neural Gaussian. Subsequently, neural Gaussians with opacity $\sigma > 0$ are rendered through a differentiable Gaussian rasterization pipeline, effectively capturing specular highlights and anisotropy in the scene.

To address these discontinuity issues, differentiable point-based rendering [14, 22, 23, 60] has been extensively explored for fitting complex geometric shapes. Notably, Zhang et al. [62] employ differentiable surface splatting and utilize a radial basis function (RBF) kernel to compute the contribution of each point to each pixel.

Recently, 3D-GS [21] has employed anisotropic 3D Gaussians, initialized from Structure from Motion (SfM), to represent 3D scenes. The innovative densification mechanism and CUDA-customized differentiable Gaussian rasterization pipeline of 3D-GS have not only achieved state-of-the-art (SOTA) rendering quality but also significantly surpassed the threshold of real-time rendering. Many concurrent works have rapidly extended 3D-GS to a variety of downstream applications, including dynamic scenes [18, 24, 31, 50, 57, 58], text-to-3D generation [10, 11, 26, 44, 59], avatars [19, 35, 39, 64, 65], and scene editing [7, 53].

Despite achieving SOTA results on commonly used benchmark datasets, 3D-GS still struggles to model scenes with specular and reflective components, which limits its practical application in real-time rendering at the photorealistic level. In this work, by replacing spherical harmonics (SH) with an anisotropic spherical Gaussian (ASG) appearance field, we have enabled 3D-GS to model complex specular scenes more effectively. Furthermore, this improvement enhances rendering quality in general scenes without significantly impacting rendering speed.

3 METHOD

The overview of our method is illustrated in Fig. 2. The input to our model is a set of posed images of a static scene, together with a sparse point cloud obtained from SfM [40]. The core of our method is to use the ASG appearance field to replace SH in modeling the appearance of 3D Gaussians (Sec. 3.2). To reduce the storage overhead and rendering speed pressure introduced by ASG, we design a hybrid Gaussian model that employs sparse anchor Gaussians to facilitate the generation of neural Gaussians (Sec. 3.3) to model the 3D scene. Finally, we introduce a simple yet effective coarse-to-fine training strategy to reduce floaters in real-world scenes (Sec. 3.4).

3.1 Preliminaries

3.1.1 3D Gaussian Splatting. 3D-GS [21] is a point-based method that employs anisotropic 3D Gaussians to represent scenes. Each 3D Gaussian is defined by a center position $\boldsymbol{x}$, opacity σ, and a 3D covariance matrix Σ, which is decomposed into a quaternion $\boldsymbol{r}$ and scaling $\boldsymbol{s}$. The view-dependent appearance of each 3D Gaussian is represented using the first three orders of spherical harmonics (SH). This method not only retains the rendering details offered by volumetric rendering but also achieves real-time rendering through a CUDA-customized differentiable Gaussian rasterization process. Following [66], the 3D Gaussians can be projected to 2D using the 2D covariance matrix Σ', defined as:

$$\Sigma' = JV\Sigma V^T J^T, \tag{1}$$

where J is the Jacobian of the affine approximation of the projective transformation, and V represents the view matrix, transitioning from world to camera coordinates. To facilitate learning, the 3D covariance matrix Σ is decomposed into two learnable components: the quaternion $\boldsymbol{r}$, representing rotation, and the 3D-vector $\boldsymbol{s}$, representing scaling. The resulting Σ is thus represented as the combination of a rotation matrix R and scaling matrix S as:

$$\Sigma = RSS^T R^T. \tag{2}$$

The color of each pixel on the image plane is then rendered through a point-based volumetric rendering (alpha blending) technique:

$$C(\mathbf{p}) = \sum_{i \in N} T_i \alpha_i c_i, \quad \alpha_i = \sigma_i e^{-\frac{1}{2}(\mathbf{p}-\mu_i)^T \Sigma'(\mathbf{p}-\mu_i)}, \tag{3}$$

where $\mathbf{p}$ denotes the pixel coordinate, T_i is the transmittance defined by $\Pi_{j=1}^{i-1}(1-\alpha_j)$, c_i signifies the color of the sorted Gaussians

associated with the queried pixel, and μ_i represents the coordinates of the 3D Gaussians when projected onto the 2D image plane.

3.1.2 Anisotropic Spherical Gaussian. Anisotropic spherical Gaussian (ASG) [54] has been designed within the traditional rendering pipeline to efficiently approximate lighting and shading. Different from spherical Gaussian (SG), ASG has been demonstrated to effectively represent anisotropic scenes with a relatively small number. In addition to retaining the fundamental properties of SG, ASG also exhibits rotational invariance and can represent full-frequency signals. The ASG function is defined as:

$$ASG(v \mid [\mathbf{x}, \mathbf{y}, \mathbf{z}], [\lambda, \mu], \xi) = \xi \cdot \mathrm{S}(v; \mathbf{z}) \cdot e^{-\lambda(v \cdot \mathbf{x})^2 - \mu(v \cdot \mathbf{y})^2}, \quad (4)$$

where v is the unit direction serving as the function input; $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ correspond to the tangent, bi-tangent, and lobe axis, respectively, and are mutually orthogonal; $\lambda \in \mathbb{R}^1$ and $\mu \in \mathbb{R}^1$ are the sharpness parameters for the $\mathbf{x}$- and $\mathbf{y}$-axis, satisfying $\lambda, \mu > 0$; $\xi \in \mathbb{R}^2$ is the lobe amplitude; S is the smooth term defined as $\mathrm{S}(v; \mathbf{z}) = \max(v \cdot \mathbf{z}, 0)$.

Inspired by the power of ASG in modeling scenes with complex anisotropy, we propose integrating ASG into Gaussian splatting to join the forces of classic models with new rendering pipelines for higher quality. For N ASGs, we predefined orthonormal axes $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, initializing them to be uniformly distributed across a hemisphere. During training, we allow the remaining ASG parameters, λ, μ, and ξ, to be learnable. We use the reflect direction w_r as the input to query ASG for modeling the view-dependent specular information. Note that we use $N = 32$ ASGs for each 3D Gaussian.

3.2 Anisotropic View-Dependent Appearance

3.2.1 ASG Appearance Field for 3D Gaussians. Although SH has enabled view-dependent scene modeling, the low frequency of low-order SH makes it challenging to model scenes with complex optical phenomena such as specular highlights and anisotropic effects. Therefore, instead of using SH, we propose using an ASG appearance field based on Eq. (4) to model the appearance of each 3D Gaussian. However, the introduction of ASG increases the feature dimensions of each 3D Gaussian, raising the model's storage overhead. To address this, we employ a compact learnable MLP Θ to predict the parameters for N ASGs, with each Gaussian carrying only additional local features $\mathbf{f} \in \mathbb{R}^{24}$ as the input to the MLP:

$$\Theta(\mathbf{f}) \rightarrow \{\lambda, \mu, \xi\}_N. \quad (5)$$

To better differentiate between high and low-frequency information and further assist ASG in fitting high-frequency specular details, we decompose color c into diffuse and specular components:

$$c = c_d + c_s, \quad (6)$$

where c_d represents the diffuse color, modeled using the first three orders of SH, and c_s is the specular color calculated through ASG. We refer to this comprehensive approach to appearance modeling as the ASG appearance field.

Although ASG theoretically enhance the ability of SH to model anisotropy, directly using ASG to represent the specular color of each 3D Gaussian still falls short in accurately modeling anisotropic and specular components, as demonstrated in Fig. 5. Inspired by [15], we do not use ASG directly to represent color but instead employ ASG to model the latent feature of each 3D Gaussian. This latent

Fig. 3. Using a coarse-to-fine strategy, our approach is able to optimize the scene in a progressive manner and eliminate the floaters efficiently.

feature, containing anisotropic information, is then fed into a tiny feature decoupling MLP Ψ to determine the final specular color:

$$\begin{aligned} &\Psi(\kappa, \gamma(\mathbf{d}), \langle n, -\mathbf{d} \rangle) \rightarrow c_s, \\ &\kappa = \bigoplus_{i=1}^{N} ASG(\omega_r \mid [\mathbf{x}, \mathbf{y}, \mathbf{z}], [\lambda_i, \mu_i], \xi_i) \end{aligned} \quad (7)$$

where κ is the latent feature derived from ASG, $\bigoplus$ denotes the concatenation operation, γ represents the positional encoding, $\mathbf{d}$ is the unit view direction pointing from the camera to each 3D Gaussian, n is the normal of each 3D Gaussian that will be discussed in Sec. 3.2.2, and ω_r is the unit reflect direction. This strategy significantly enhances the ability of 3D-GS to model scenes with complex optical phenomena, whereas neither pure ASG nor pure MLP can achieve anisotropic appearance modeling as effectively as our approach.

3.2.2 Normal Estimation. Directly estimating the normals of 3D Gaussians presents a challenge, as 3D-GS comprises a collection of discrete entities, each representing a local space within a certain range, without forming a continuous surface. The calculation of normals typically necessitates a continuous surface, and the anisotropic shape of each entity in 3D-GS further complicates the determination of normals. Following [20, 41], we use the shortest axis of each Gaussian as its normal. This approach is based on the observation that 3D Gaussians tend to flatten gradually during the optimization process, allowing the shortest axis to serve as a reasonable approximation for the normal.

The reflect direction ω_r can then be derived using the view direction and the local normal vector n as:

$$\omega_r = 2(\omega_o \cdot n) \cdot n - \omega_o, \quad (8)$$

where $\omega_o = -\mathbf{d}$ is a unit view direction pointing from each 3D Gaussian in world space to the camera. We use the reflect direction ω_r to query ASG, enabling better interpolation of latent features containing anisotropic information. Experimental results show that although this unsupervised normal estimation cannot generate physically accurate normals aligned with the real world, it is sufficient to produce relatively accurate reflect direction to assist ASG in fitting high-frequency information.

3.3 Anchor-Based Gaussian Splatting

3.3.1 Neural Gaussian Derivation with ASG Appearance Field. While the ASG appearance field significantly improves the ability of 3D-GS to model specular and anisotropic features, it introduces additional storage and computational overhead compared to using pure SH due to the additional local features $\mathbf{f}$ associated with each Gaussian. Although real-time rendering at over 100 FPS is still achievable in bounded scenes, the substantial increase in storage overhead and reduction in rendering speed caused by ASG in real-world unbounded scenes is unacceptable. Inspired by [30], we employ anchor-based Gaussian splatting to reduce storage overhead and the number of 3D Gaussians required for rendering, thereby accelerating the rendering.

Unlike the attributes carried by each entity in 3D-GS, each anchor Gaussian carries a position coordinate $\mathbf{P}_v \in \mathbb{R}^3$, a local feature $\mathbf{f}_v \in \mathbb{R}^{32}$, a displacement factor $\eta_v \in \mathbb{R}^3$, and k learnable offsets $\mathbf{O}_v \in \mathbb{R}^{k\times 3}$. We use the sparse point cloud obtained from COLMAP [40] to initialize each anchor 3D Gaussian, serving as the voxel centers to guide the generation of neural Gaussians. The position $\mathbf{P}_v$ of the anchor Gaussian is initialized as:

$$\mathbf{P}_v = \left\{ \left\lfloor \frac{\mathbf{P}}{\epsilon} + 0.5 \right\rfloor \right\} \cdot \epsilon, \tag{9}$$

where $\mathbf{P}$ is the position of point cloud, ϵ denotes the voxel size, and $\{\cdot\}$ denotes removing duplicated anchors.

We then use the anchor Gaussians to guide the generation of neural Gaussians, which have the same attributes as vanilla 3D-GS. For each visible anchor Gaussian within the viewing frustum, we spawn k neural Gaussians and predict their attributes (see Fig. 2). The positions $\mathbf{x}$ of neural Gaussians are calculated as:

$$\{\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}\} = \mathbf{P}_v + \{\mathbf{O}_0, \ldots, \mathbf{O}_{k-1}\} \cdot \eta_v, \tag{10}$$

where $\mathbf{P}_v$ represents the position of the anchor Gaussian corresponding to k neural Gaussians. The opacity σ is calculated through a tiny MLP:

$$\{\sigma_0, \ldots, \sigma_{k-1}\} = \mathcal{F}_\sigma\left(\mathbf{f}_v, \delta_{cv}, \mathbf{d}_{cv}\right), \tag{11}$$

where δ_{cv} denotes the distance between the anchor Gaussian and the camera, and $\mathbf{d}_{cv}$ is the unit direction pointing from the camera to the anchor Gaussian. The rotation r and scaling s of each neural Gaussian are derived similarly using the corresponding tiny MLP $\mathcal{F}_r$ and $\mathcal{F}_s$.

Since the anisotropy modeled by ASG is continuous in space, it can be compressed into a lower-dimensional space. Thanks to the guidance of the anchor Gaussian, the anchor feature $\mathbf{f}_v$ can be used directly to compress N ASGs, further reducing storage pressure. To make the ASG of neural Gaussians position-aware, we introduce the unit view direction to decompress ASG parameters. Consequently, the ASG parameters prediction in Eq. (5) is revised as follows:

$$\Theta(\mathbf{f}_\mathbf{v}, \mathbf{d}_{cn}) \rightarrow \{\lambda, \mu, \xi\}_N, \tag{12}$$

where $\mathbf{d}_{cn}$ denotes the unit view direction from the camera to each neural Gaussian. Additionally, we set the diffuse part of the neural Gaussian $c_d = \phi(\mathbf{f}_v)$, directly predicted through an MLP ϕ, to ensure the smoothness of the diffuse component and reduce the difficulty of convergence.

3.3.2 Adaptive Control of Anchor Gaussians. To enable 3D-GS to represent scene details while removing redundant entities, we adaptively adjust the number of anchor Gaussians based on the gradient and opacity of the neural Gaussians. Following [21, 30], we compute the averaged gradients of the k spawned neural Gaussians every 100 training iterations for each anchor Gaussian, denoted as ∇_v. Anchor Gaussians with $\nabla_v > \tau_g$ will be densified. In practice, we follow [30] to quantize the space into multi-resolution voxels to allow new anchor Gaussians to be added at different granularities:

$$\epsilon^{(l)} = \epsilon \cdot \beta/4^l, \quad \tau_g^{(l)} = \tau_g \cdot 2^l, \tag{13}$$

where l denotes the level of new anchor Gaussians, $\epsilon^{(l)}$ is the voxel size at the l-th level for newly grown anchor Gaussians, and β represents a growth factor. Different from [30], to reduce overfitting caused by excessive densification of anchors, we introduced a hierarchical selection. Only anchor Gaussians with $\nabla_v > \text{Quantile}(\nabla_v, 2^{-(l+1)})$ will be densified at the corresponding voxel center at the l-th level.

To eliminate trivial anchors, we accumulate the opacity values of their associated neural Gaussians for every 100 training iteration, denoted as $\bar{\sigma}$. If an anchor Gaussian fails to produce neural Gaussians with a satisfactory level of opacity, with $\bar{\sigma} < \tau_o$, we remove it.

3.4 Coarse-to-fine Training

We observed that in many real-world scenarios, 3D-GS tends to overfit the training data, leading to the emergence of numerous floaters when rendering images from novel viewpoints. A common challenge in real-world datasets is inaccuracies in camera pose estimation, particularly evident in large scenes. Scaffold-GS [30], by anchoring 3D-GS, imposes a sparse voxel constraint on the geometry, creating a hierarchical 3D-GS representation. While this hierarchical approach improves the ability of 3D-GS to model complex geometries, it does not address the overfitting issue and, in many cases, exacerbates the presence of floaters in scene backgrounds.

To mitigate the occurrence of floaters in real-world scenes, we propose a coarse-to-fine training mechanism. We believe that the tendency of 3D-GS to overfit stems from an excessive focus on each 3D Gaussian's contribution to a specific pixel and its immediate neighbors, rather than considering broader global information. Therefore, we decide to train 3D-GS progressively from low to high resolution:

$$r(i) = \min(\lfloor r_s + (r_e - r_s) \cdot i/\tau \rceil, r_e), \tag{14}$$

where $r(i)$ is the image resolution at the i-th training iteration, r_s is the starting image resolution, r_e is the ending image resolution (the full resolution we aim to render), and τ is the threshold iteration, empirically set to 20k.

This training approach enables 3D-GS to learn global information from the images in the early stages of training, thereby reducing overfitting to local areas of the training images and eliminating a significant number of floaters in novel view rendering. Additionally, due to the lower resolution training in the initial phase, this mechanism reduces training time by approximately 20%.

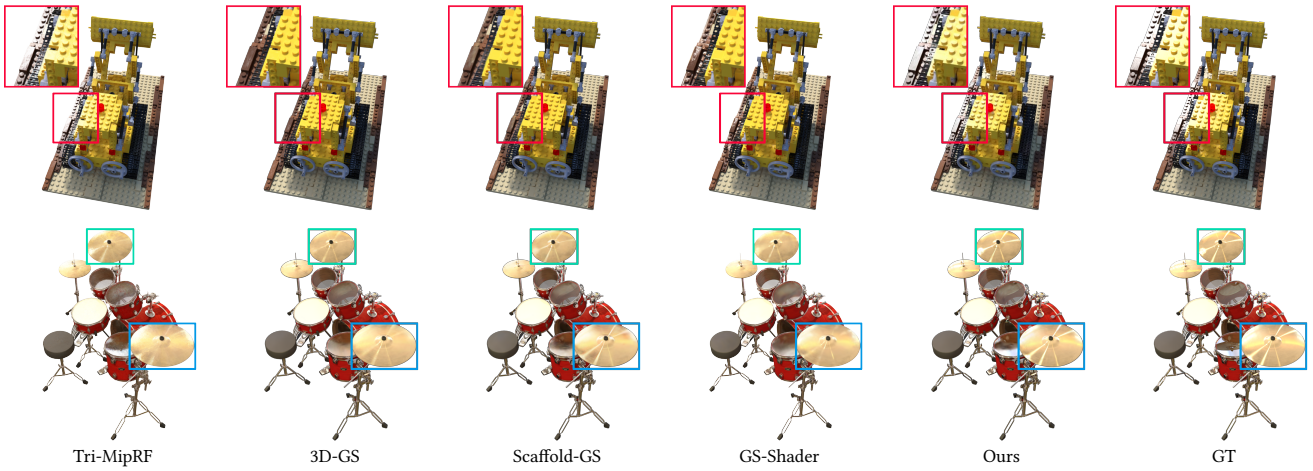

Fig. 4. **Visualization on NeRF dataset.** Our method has successfully achieved local specular highlights modeling, a capability that other 3D-GS-based methods fail to accomplish, while maintaining fast rendering speed. Compared to Tri-MipRF, a NeRF-based method, we have significantly enhanced the ability to model anisotropic materials.

Dataset	Mip-NeRF360					Tanks&Temples					Deep Blending				
Method | Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem	PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem	PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem
Mip-NeRF360	27.69	0.792	0.237	0.06	8.6MB	22.22	0.759	0.257	0.14	8.6MB	29.40	0.901	0.245	0.09	8.6MB
iNGP	25.59	0.699	0.331	9.43	48MB	21.72	0.723	0.330	14.4	48MB	23.62	0.797	0.423	2.79	48MB
Plenoxels	23.08	0.626	0.463	6.79	2.1GB	21.08	0.719	0.379	13.0	2.3GB	23.06	0.795	0.510	11.2	2.7GB
3D-GS	27.47	0.812	0.222	115	748MB	23.71	0.844	0.178	169	432MB	29.65	0.899	0.247	130	662MB
Scaffold-GS	27.66	0.807	0.236	96	203MB	23.96	0.853	0.177	143	89.5MB	30.21	0.906	0.254	179	63.5MB
Ours-w/o anchor	27.81	0.810	0.223	25	1.02GB	23.94	0.846	0.181	37	563MB	29.71	0.901	0.250	32	793MB
Ours	28.01	0.812	0.222	70	245MB	24.58	0.855	0.174	111	96.5MB	30.45	0.906	0.252	132	68MB

Table 1. **Quantitative evaluation of our method compared to previous work on real-world datasets.** We report PSNR, SSIM, LPIPS(VGG) and color each cell as best , second best and third best . Our method has overall achieved the best rendering quality, while also striking a good balance between FPS and the storage memory of 3D Gaussians.

3.5 Losses

In addition to the color loss in 3D-GS [21], we also incorporate a regularization loss to encourage the neural Gaussians to remain small and minimally overlapping. Consequently, the total loss function for all learnable parameters and MLPs is formulated as:

$$\mathcal{L} = (1 - \lambda_{\text{D-SSIM}})\mathcal{L}_1 + \lambda_{\text{D-SSIM}}\mathcal{L}_{\text{D-SSIM}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}},$$
$$\mathcal{L}_{\text{reg}} = \frac{1}{N}\sum_{i=1}^{N_{\text{n}}} \text{Prod}\left(s_i\right), \tag{15}$$

where N_n is the number of neural Gaussians and $\text{Prod}(\cdot)$ calculates the product of the scale s_i of each neural Gaussian. The $\lambda_{\text{D-SSIM}} = 0.2$ and $\lambda_{\text{reg}} = 0.01$ are consistently used in our experiments.

4 EXPERIMENTS

In this section, we present both quantitative and qualitative results of our method. To evaluate its effectiveness, we compared it to several state-of-the-art methods across various datasets. We color each cell as best , second best and third best . Our method demonstrates superior performance in modeling complex specular and anisotropic features, as evidenced by comparisons on the NeRF, NSVF, and our "Anisotropic Synthetic" datasets. Additionally, we showcase its versatility by comparing its performance across all scenarios in 3D-GS, further proving the robustness of our approach.

Dataset	NeRF Synthetic				
Method | Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem
iNGP-Base	33.18	0.963	0.045	~10	~13MB
Mip-NeRF	33.09	0.961	0.043	<1	~10MB
Tri-MipRF	33.65	0.963	0.042	~5	~60MB
3D-GS	33.32	0.970	0.031	415	69MB
GS-Shader	33.38	0.968	0.029	97	29MB
Scaffold-GS	33.68	0.967	0.034	240	19MB
Ours-w anchor	33.96	0.969	0.032	162	20MB
Ours	34.12	0.971	0.028	105	79MB

Table 2. **Quantitative results on NeRF synthetic dataset.** Our method achieves a rendering quality that surpasses NeRF-based methods, without excessively reducing FPS.

Dataset	NSVF Synthetic				
Method | Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem
TensoRF	36.52	0.982	0.026	~1.5	~65MB
Tri-MipRF	34.58	0.973	0.030	~5	~60MB
NeuRBF	37.80	0.986	0.019	~1	~580MB
3D-GS	37.07	0.987	0.015	403	66MB
GS-Shader	33.85	0.981	0.020	68	33MB
Scaffold-GS	36.43	0.984	0.017	218	17MB
Ours-w anchor	37.71	0.987	0.015	142	18MB
Ours	38.35	0.988	0.013	91	99MB

Table 3. **Quantitative results on NSVF synthetic dataset.** Our method achieved significantly higher rendering quality than 3D-GS, and it also surpassed NeRF-based methods.

Dataset	Anisotropic Synthetic				
Method | Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem
3D-GS	33.82	0.966	0.062	345	47MB
Scaffold-GS	35.34	0.972	0.052	234	27MB
Ours-w anchor	36.76	0.976	0.046	180	28MB
Ours	37.28	0.977	0.047	119	64MB

Table 4. **Quantitative results on our "Anisotropic Synthetic" dataset.**

4.1 Implementation Details

We implemented our framework using PyTorch [37] and modified the differentiable Gaussian rasterization to include depth visualization. For the ASG appearance field, the feature decoupling MLP Ψ consists of 3 layers, each with 64 hidden units, and the positional encoding for the view direction is of order 2. In terms of anchor-based Gaussian splatting, we established three levels for anchor Gaussian densification, setting the densification threshold τ_g to 0.0002, the pruning threshold τ_o to 0.005, and the number of neural Gaussians of each anchor k to 10. In bounded scenes, our voxel size ϵ is set to 0.001 with a growth factor β of 4. For the Mip-360 scenes, the voxel size remains 0.001, but the growth factor β is increased to 16. Regarding coarse-to-fine training, we start with a resolution r_s that is **8x** downsampled. To further accelerate rendering speed, we prefilter the visible anchor Gaussians and allow only those neural Gaussians with opacity $\sigma_n > 0$ to pass through the ASG appearance field and Gaussian rasterization pipelines. All experiments were conducted on a Tesla V100, and FPS measurements were performed on an NVIDIA RTX 3090 with 24GB of memory.

4.2 Results and Comparisons

Synthetic Bounded Scenes. We used the NeRF, NSVF, and our "Anisotropic Synthetic" datasets as the experimental datasets for synthetic scenes. Our comparisons were made with the most relevant state-of-the-art methods, including 3D-GS [21], Scaffold-GS [30], GaussianShader [20], and several NeRF-based methods such as NSVF [28], TensoRF [5], NeuRBF [9], and Tri-MipRF [17]. To ensure a fair comparison, we used the rendering metrics of the NeRF and NSVF datasets as reported in the baseline papers. For scenes not reported in the baseline papers, we trained the baselines from scratch using the released codes and their default configurations.

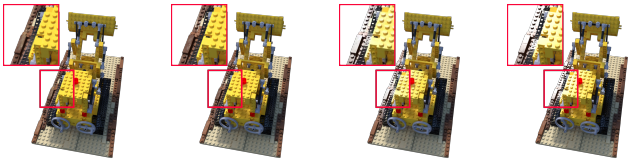

Fig. 5. **Ablation on ASG feature decoupling MLP.** We show that directly using ASG to model color leads to the failure in modeling anisotropy and specular highlights. By decoupling the ASG features through MLP, we can realistically model complex optical phenomena.

As shown in Fig. 4, Figs. 7-8. and Tabs. 2-4, our method achieved the highest performance in terms of PSNR, SSIM, and LPIPS. It also significantly improved upon the issues that 3D-GS faced in modeling high-frequency specular highlights and complex anisotropy. See per-scene results in the supplementary materials.

Real-world Unbounded Scenes. To verify the versatility of our method in real-world scenarios, we used the same real-world dataset as in 3D-GS [21]. As shown in Tab. 1, our method achieves rendering results comparable to state-of-the-art methods on the Deep Blending dataset and surpasses them on Mip-NeRF 360 and Tanks&Temples. Furthermore, our method effectively balances FPS, storage overhead, and rendering quality. It enhances rendering quality without excessively increasing storage requirements or significantly reducing FPS. As illustrated in Fig. 6, our method has also significantly improved the visual effect. It removes a large number of floaters in outdoor scenes and successfully models the high-frequency specular highlights in indoor scenes. This demonstrates that our approach is not only adept at modeling complex specular scenes but also effectively improves rendering quality in general scenarios.

4.3 Ablation Study

4.3.1 ASG feature decoupling MLP. We conducted an ablation study to evaluate the effectiveness of using ASG to output features, which are then decoupled through an MLP Ψ to derive the final specular color. As demonstrated in Fig. 5, directly using ASG to output color results in the inability to model specular and anisotropic components. In contrast to directly using an MLP for color modeling, as in Scaffold-GS [30], ASG can encode higher-frequency anisotropic features. This capability aids the MLP in learning complex optical phenomena, leading to more accurate and detailed rendering results.

4.3.2 Coarse-to-fine training. We conducted an ablation study to assess the impact of coarse-to-fine (c2f) training. As illustrated in Fig. 10, both 3D-GS and Scaffold-GS exhibit a large number of floaters in the novel view synthesis. Coarse-to-fine training effectively reduces the number of floaters, alleviating the overfitting issue commonly encountered by 3D-GS in real-world scenarios.

5 CONCLUSION

In this work, we introduce *Spec-Gaussian*, a novel approach to 3D Gaussian splitting that features an anisotropic view-dependent appearance. Leveraging the powerful capabilities of ASG, our method

effectively overcomes the challenges encountered by vanilla 3D-GS in rendering scenes with specular highlights and anisotropy. Additionally, we innovatively implement a coarse-to-fine training mechanism to eliminate floaters in real-world scenes. Both quantitative and qualitative experiments demonstrate that our method not only equips 3D-GS with the ability to model specular highlights and anisotropy but also enhances the overall rendering quality of 3D-GS in general scenes, without significantly compromising FPS and storage overhead.

Limitations. Although our method enables 3D-GS to model complex specular and anisotropic features, it still faces challenges in handling reflections. Specular and anisotropic effects are primarily influenced by material properties, whereas reflections are closely related to the environment and geometry. Due to the lack of explicit geometry in 3D-GS, we cannot differentiate between reflections and material textures using constraints like normals, as employed in Ref-NeRF [45] and NeRO [29]. In our experiments, we also observed that when ground truth geometric information is provided, 3D-GS becomes more consistent with expectations under strict constraints, but this comes at the cost of a certain decline in rendering quality. We plan to explore solutions for modeling reflections with 3D-GS in future work.

REFERENCES

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

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

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. *ICCV* (2023).

[4] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. 2005. High-quality surface splatting on today's GPUs. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* IEEE, 17–141.

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*. Springer, 333–350.

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision (ECCV)*.

[7] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. 2023. GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting. arXiv:2311.14521 [cs.CV]

[8] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2022. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277* (2022).

[9] Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. 2023. Neurbf: A neural fields representation with adaptive radial basis functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4182–4194.

[10] Zilong Chen, Feng Wang, and Huaping Liu. 2023. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585* (2023).

[11] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. 2023. LucidDreamer: Domain-free Generation of 3D Gaussian Splatting Scenes. *arXiv preprint arXiv:2311.13384* (2023).

[12] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14346–14355.

[14] Markus Gross and Hanspeter Pfister. 2011. *Point-based graphics*. Elsevier.

[15] Kang Han and Wei Xiang. 2023. Multiscale Tensor Decomposition and Rendering Equation Encoding for View Synthesis. In *The IEEE / CVF Computer Vision and Pattern Recognition Conference*. 4232–4241.

[16] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. 2021. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5875–5884.

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

[18] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2023. SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. *arXiv preprint arXiv:2312.14937*, 1–11.

[19] Yuheng Jiang, Zhehao Shen, Penghao Wang, Zhuo Su, Yu Hong, Yingliang Zhang, Jingyi Yu, and Lan Xu. 2023. HiFi4G: High-Fidelity Human Performance Rendering via Compact Gaussian Splatting. *arXiv preprint arXiv:2312.03461* (2023).

[20] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. 2023. GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces. *arXiv preprint arXiv:2311.17977* (2023).

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[22] Leonid Keselman and Martial Hebert. 2022. Approximate differentiable rendering with algebraic surfaces. In *European Conference on Computer Vision*. Springer, 596–614.

[23] Leonid Keselman and Martial Hebert. 2023. Flexible techniques for differentiable rendering with 3d gaussians. *arXiv preprint arXiv:2308.14737* (2023).

[24] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2023. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. *arXiv preprint arXiv:2312.16812* (2023).

[25] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. 2023. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[26] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. 2023. LucidDreamer: Towards High-Fidelity Text-to-3D Generation via Interval Score Matching. arXiv:2311.11284 [cs.CV]

[27] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. 2021. BARF: Bundle-Adjusting Neural Radiance Fields. In *IEEE International Conference on Computer Vision (ICCV)*.

[28] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* 33 (2020), 15651–15663.

[29] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. 2023. NeRO: Neural Geometry and BRDF Reconstruction of Reflective Objects from Multiview Images. In *SIGGRAPH*.

[30] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. 2023. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. *arXiv preprint arXiv:2312.00109* (2023).

[31] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *3DV*.

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530127

[34] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8280–8290.

[35] Haokai Pang, Heming Zhu, Adam Kortylewski, Christian Theobalt, and Marc Habermann. 2023. ASH: Animatable Gaussian Splats for Efficient and Photoreal Human Rendering. (2023). arXiv:2312.05941 [cs.CV]

[36] Keunhong Park, Philipp Henzler, Ben Mildenhall, Jonathan T Barron, and Ricardo Martin-Brualla. 2023. CamP: Camera preconditioning for neural radiance fields. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–11.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).

[38] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14335–14345.

[39] Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. 2023. Relightable Gaussian Codec Avatars. (2023). arXiv:2312.03704 [cs.GR]
[40] Johannes L Schonberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*. 4104–4113.
[41] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jingtuo Liu, Liangjun Zhang, Jian Zhang, Bin Zhou, et al. 2023. GIR: 3D Gaussian Inverse Rendering for Relightable Scene Factorization. *arXiv preprint arXiv:2312.05133* (2023).
[42] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. 2021. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. In *CVPR*.
[43] Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5459–5469.
[44] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. 2023. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. *arXiv preprint arXiv:2309.16653* (2023).
[45] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. *CVPR* (2022).
[46] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *NeurIPS* (2021).
[47] Peng Wang, Lingzhe Zhao, Ruijie Ma, and Peidong Liu. 2023. BAD-NeRF: Bundle Adjusted Deblur Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4170–4179.
[48] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021. NeRF−−: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064* (2021).
[49] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. NeX: Real-time View Synthesis with Neural Basis Expansion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[50] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 2023. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. *arXiv preprint arXiv:2310.08528* (2023).
[51] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Holynski. 2023. ReconFusion: 3D Reconstruction with Diffusion Priors. *arXiv* (2023).
[52] Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. 2023. Voxurf: Voxel-based Efficient and Accurate Neural Surface Reconstruction. In *International Conference on Learning Representations (ICLR)*.
[53] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. 2023. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. *arXiv preprint arXiv:2311.12198* (2023).
[54] Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. 2013. Anisotropic Spherical Gaussians. *ACM Transactions on Graphics* 32, 6 (2013), 209:1–209:11.
[55] Jiawei Yang, Marco Pavone, and Yue Wang. 2023. FreeNeRF: Improving Few-shot Neural Rendering with Free Frequency Regularization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[56] Ziyi Yang, Yanzhen Chen, Xinyu Gao, Yazhen Yuan, Yu Wu, Xiaowei Zhou, and Xiaogang Jin. 2023. SIRe-IR: Inverse Rendering for BRDF Reconstruction with Shadow and Illumination Removal in High-Illuminance Scenes. *arXiv preprint arXiv:2310.13030* (2023).
[57] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2023. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. *arXiv preprint arXiv:2309.13101* (2023).
[58] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. 2023. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. *arXiv preprint arXiv 2310.10642* (2023).
[59] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. 2023. GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models. *arXiv preprint arXiv:2310.08529* (2023).
[60] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.
[61] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5752–5761.
[62] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. 2022. Differentiable Point-Based Radiance Fields for Efficient View Synthesis. *arXiv preprint arXiv:2205.14330* (2022).
[63] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)* 40, 6 (2021), 1–18.
[64] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. 2023. GPS-Gaussian: Generalizable Pixel-wise 3D Gaussian Splatting for Real-time Human Novel View Synthesis. *arXiv* (2023).
[65] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. 2023. Drivable 3D Gaussian Avatars. (2023). arXiv:2311.08581 [cs.CV]
[66] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001. EWA volume splatting. In *Proceedings Visualization, 2001. VIS'01.* IEEE, 29–538.

A MORE RESULTS

In this section, we present the complete quantitative results of our experiments. We report PSNR, SSIM, LPIPS (VGG), and color each cell as best, second best and third best.

A.1 NeRF Synthetic Scenes

As shown in Tabs. 5-7, our method demonstrates the best rendering quality metrics in almost every scene. It's important to note that the experimental setup for Tri-MipRF [17] differs from other methods. It uses both the training and validation sets as training data, expanding the scale of the model's data. When its training data is limited to the training set, its metrics suffer a noticeable drop. Nevertheless, to ensure that the experimental results fully reflect the highest performance of each method, and to prevent significant drops in metrics due to differences in experimental environments, we still present the metrics from the Tri-MipRF official paper. Our method achieved more prominent metrics in scenes with notable specular reflection and anisotropy, such as Drums, Lego, and Ship. This demonstrates that our method not only improves the overall rendering quality but also has a more significant advantage in complex specular scenarios.

A.2 NSVF Synthetic Scenes

The NSVF [28] dataset, in comparison to NeRF, features more noticeable metallic specular reflection, as presented in the Wineholder, Steamtrain, and Spaceship scenes. It also includes more complex transmission scenarios, such as Lifestyle. It is important to note that Tri-MipRF fails to converge in the Steam scene with the official code, so we did not report metrics for that scenario. As shown in Tabs. 8-10, we present the per-scene experimental results of PSNR, SSIM, and LPIPS in the supplementary material. The experimental results indicate that compared to other methods based on 3D-GS [21], our method has significant advantages in metallic highlights and complex transmission scenarios. Additionally, we compared it with the SOTA NeRF-based methods based on NeRF. Our approach enables 3D-GS to surpass the latest SOTA of NeRF, achieving high-frequency highlight modeling that 3D-GS couldn't realize but NeRF could, thereby achieving truly high-quality rendering.

A.3 Anisotorpic Synthetic Scenes

"Anisotropic Synthetic" is a synthetic dataset we rendered ourselves, which includes 8 scenes with significant anisotropy. We tested some existing 3D-GS-based methods on "Anisotropic Synthetic." As shown in Tabs. 11-13, our method achieved a very significant improvement in rendering metrics. Fig. 11 shows the comparison between our method and 3D-GS across all eight scenes. Qualitative experiments

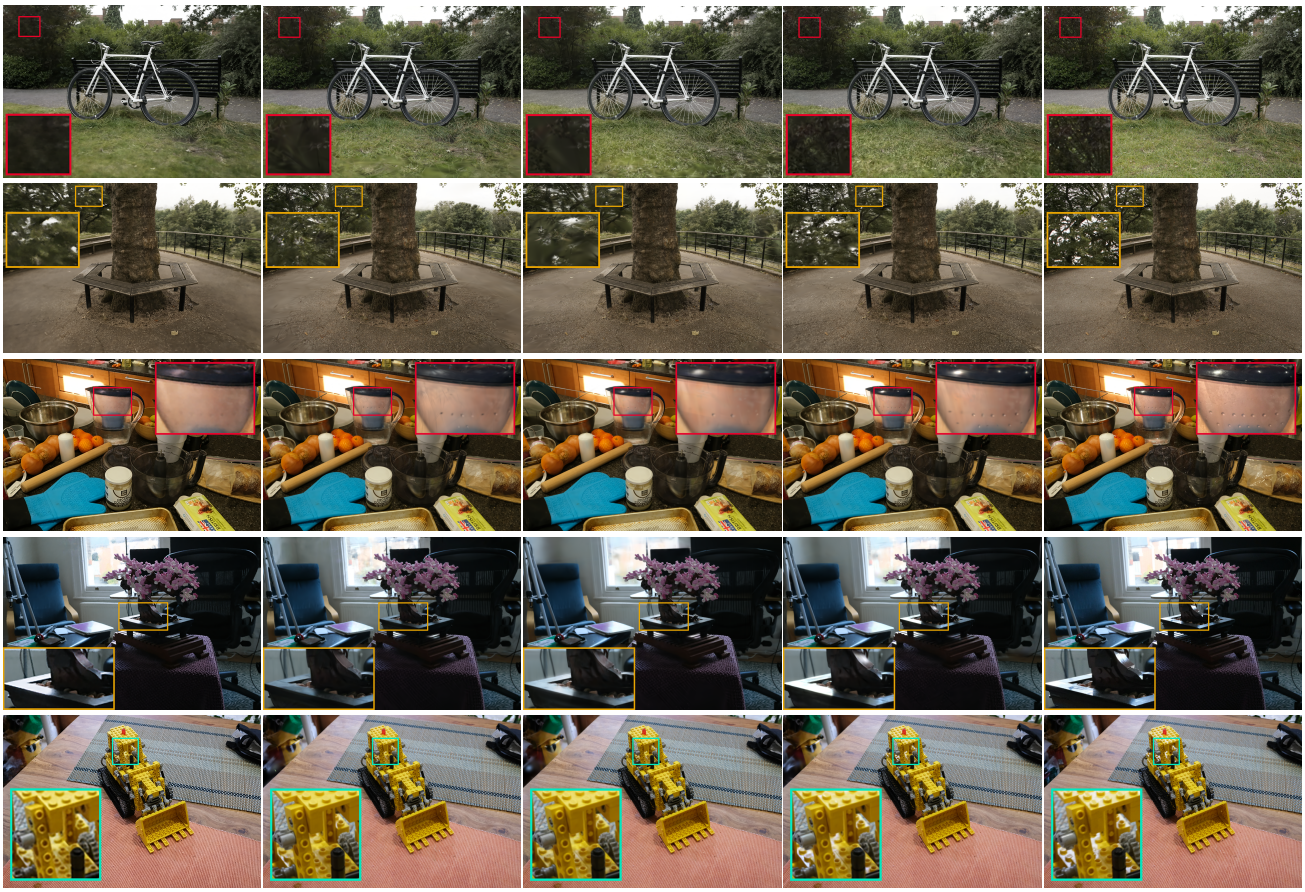

Fig. 6. **Visualization on Mip-NeRF 360 dataset.** This clearly demonstrates that our method is capable of modeling complex specular highlights and effectively removing floaters, outperforming other methods in these aspects.

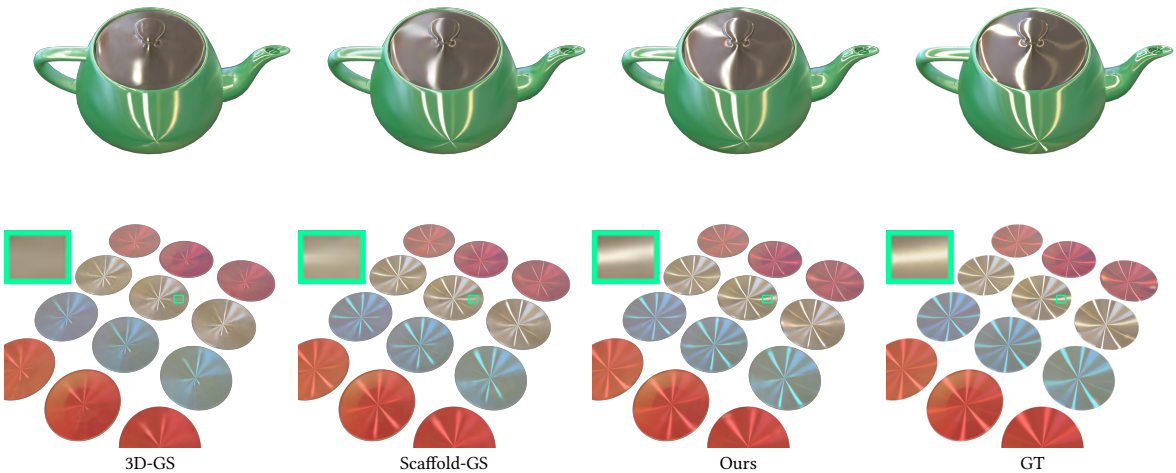

Fig. 7. **Visualization on our anisotropic dataset.** We have demonstrated the superiority of our method compared to 3D-GS and scaffold-GS, which models color based on MLP. With the help of ASG, we can model specular highlights and anisotropic parts of the scene more effectively.

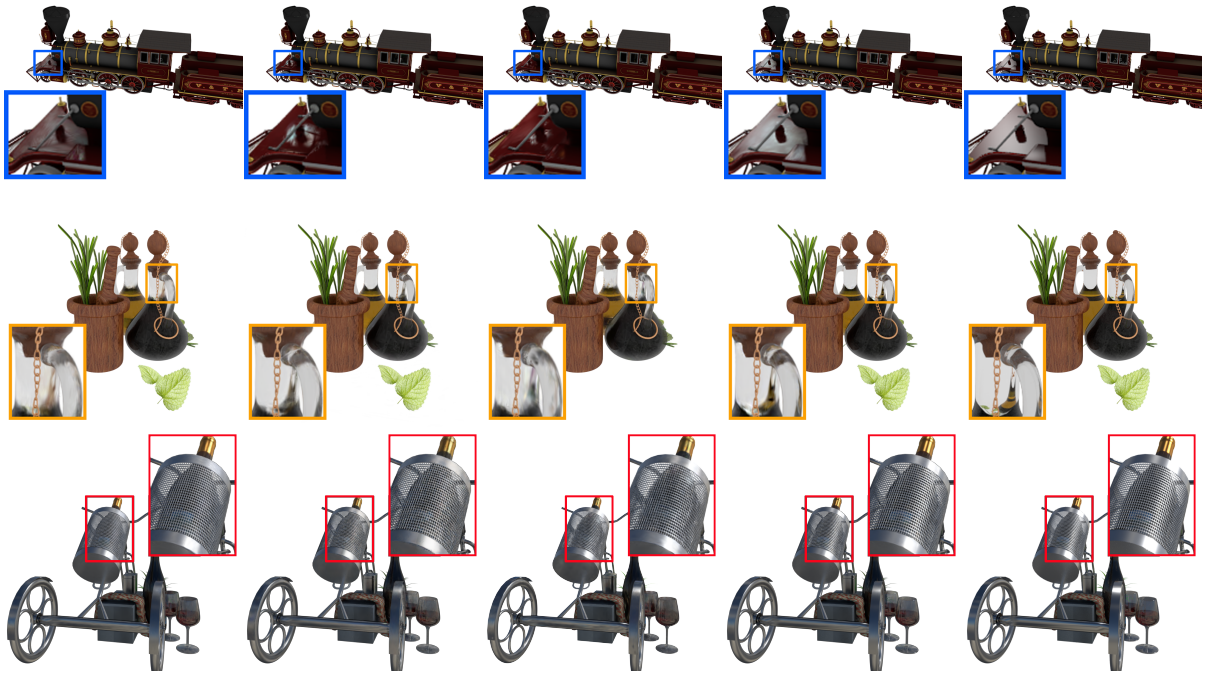

Fig. 8. **Visualization on NSVF dataset.** Our method significantly improves the ability to model metallic materials compared to other GS-based methods. At the same time, our method also demonstrates the capability to model refractive parts, reflecting the powerful fitting ability of our method.

Fig. 9. **Ablation on anchor Gaussians.** The shiny scene is borrowed from Nex [49]. This clearly demonstrates that anchor Gaussians can improve the geometry of 3D-GS. Consequently, this enhancement aids in its ability to learn the reflective parts of the scene, as highlighted in the orange and blue boxes.

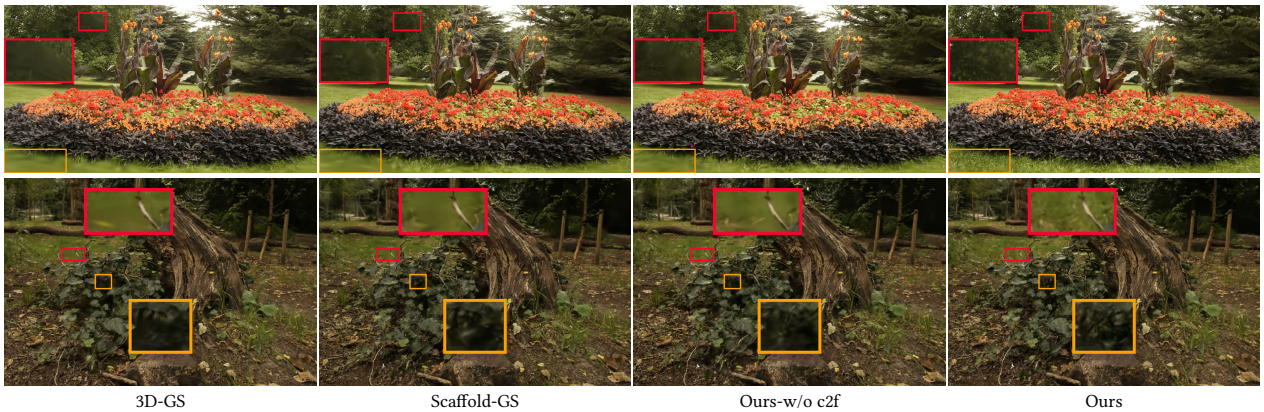

Fig. 10. **Ablation on coarse-to-fine training mechanism.** Experimental results demonstrate that our simple yet effective training mechanism can effectively remove floaters in both the background and foreground, thereby alleviating the overfitting problem prevalent in 3D-GS-based methods.

also demonstrate the significant visual advantages of our method, highlighting the substantial improvement our method brings to anisotropic parts, thereby enhancing the overall rendering quality.

A.4 Mip-360 Scenes

The MipNeRF-360 scenes include five outdoor and four indoor scenarios. There are several scenes rich in specular reflections, such as bonsai, room, and kitchen. As shown in Tabs. 14-16, our method achieved significant advantages in the four indoor scenes. This reflects our method's strengths in modeling specular reflections and anisotropy. In outdoor scenes, our method also achieved rendering metrics comparable to the SOTA methods. Furthermore, with the help of the coarse-to-fine training mechanism, our method significantly reduced the number of floaters, resulting in a substantial improvement in visual effects.

A.5 Tanks & Temples and Deep Blending Scenes

To more comprehensively demonstrate the superiority of our method over 3D-GS in terms of rendering quality, we also compared it with the Deep Blending and Tanks & Temples datasets, which were also used in the original 3D-GS paper. As shown in Tab. 17, our method achieved the best metrics in almost all scenes. This also showcases the versatility of our method, indicating that it is not limited to modeling anisotropic and specular scenarios.

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Avg.
iNGP-Base	35.00	26.02	33.51	37.40	36.39	29.78	36.22	31.10	33.18
Mip-NeRF	35.14	25.48	33.29	37.48	35.70	30.71	36.51	30.41	33.09
Tri-MipRF	36.10	26.59	34.51	38.54	36.15	30.73	37.75	28.78	33.65
GS-Shader	35.83	26.36	34.97	37.85	35.87	30.07	35.23	30.82	33.38
3D-GS	35.36	26.15	34.87	37.72	35.78	30.00	35.36	30.80	33.32
Scaffold-GS	35.28	26.44	35.21	37.73	35.69	30.65	37.25	31.17	33.68
Ours-w anchor	35.57	26.58	35.71	38.12	36.62	30.66	36.81	31.63	33.96
Ours	35.68	26.92	36.14	38.28	36.07	30.85	37.12	31.89	34.12

Table 5. **Per-scene PSNR comparison on the NeRF dataset.**

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Avg.
iNGP-Base	0.979	0.937	0.981	0.982	0.982	0.951	0.990	0.896	0.963
Mip-NeRF	0.981	0.932	0.980	0.982	0.978	0.959	0.991	0.882	0.961
Tri-MipRF	0.985	0.939	0.983	0.984	0.982	0.953	0.992	0.879	0.963
GS-Shader	0.987	0.949	0.985	0.985	0.983	0.960	0.991	0.905	0.968
3D-GS	0.988	0.955	0.987	0.985	0.983	0.960	0.992	0.907	0.970
Scaffold-GS	0.985	0.950	0.985	0.983	0.980	0.960	0.992	0.898	0.967
Ours-w anchor	0.986	0.953	0.987	0.985	0.982	0.962	0.992	0.904	0.969
Ours	0.987	0.958	0.988	0.985	0.982	0.963	0.993	0.909	0.971

Table 6. **Per-scene SSIM comparison on the NeRF dataset.**

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Avg.
iNGP-Base	0.022	0.071	0.023	0.027	0.017	0.060	0.010	0.132	0.045
Mip-NeRF	0.021	0.065	0.020	0.027	0.021	0.040	0.009	0.138	0.043
Tri-MipRF	0.016	0.066	0.020	0.021	0.016	0.052	0.008	0.136	0.042
GS-Shader	0.012	0.040	0.013	0.019	0.014	0.033	0.006	0.098	0.029
3D-GS	0.011	0.037	0.012	0.020	0.016	0.037	0.006	0.106	0.031
Scaffold-GS	0.013	0.042	0.013	0.023	0.019	0.040	0.008	0.114	0.034
Ours-w anchor	0.013	0.038	0.012	0.022	0.016	0.037	0.007	0.112	0.032
Ours	0.011	0.032	0.011	0.019	0.014	0.032	0.006	0.104	0.028

Table 7. **Per-scene LPIPS (VGG) comparison on the NeRF dataset.**

	Bike	Life	Palace	Robot	Space	Steam	Toad	Wine	Avg.
NeRF	31.77	31.08	31.76	28.69	34.66	30.84	29.42	28.23	30.81
NSVF	37.75	34.60	34.05	35.24	39.00	35.13	33.25	32.04	35.13
TensoRF	39.23	34.51	37.56	38.26	38.60	37.87	34.85	31.32	36.52
Tri-MipRF	36.98	33.98	36.55	33.49	37.60	-	33.48	29.97	34.58
NeuRBF	40.71	36.08	38.93	39.13	40.44	38.35	35.73	32.99	37.80
3D-GS	40.76	33.19	38.89	39.16	36.80	37.67	37.33	32.76	37.07
Scaffold-GS	39.87	35.00	38.53	37.92	34.36	37.12	36.29	32.32	36.43
GS-Shader	37.38	27.36	36.55	37.00	32.61	35.27	34.50	30.16	33.85
Ours-w anchor	40.63	35.56	38.95	38.52	39.47	37.98	36.55	34.04	37.71
Ours	41.52	36.13	39.10	39.60	40.02	38.28	37.43	34.73	38.35

Table 8. **Per-scene PSNR comparison on the NSVF dataset.**

	Bike	Life	Palace	Robot	Space	Steam	Toad	Wine	Avg.
NeRF	0.970	0.946	0.950	0.960	0.980	0.966	0.920	0.920	0.952
NSVF	0.991	0.971	0.969	0.988	0.991	0.986	0.968	0.965	0.979
TensoRF	0.993	0.968	0.979	0.994	0.989	0.991	0.978	0.961	0.982
Tri-MipRF	0.990	0.962	0.973	0.985	0.986	-	0.968	0.945	0.973
NeuRBF	0.995	0.977	0.985	0.995	0.993	0.993	0.983	0.972	0.986
3D-GS	0.994	0.979	0.983	0.994	0.991	0.993	0.985	0.975	0.987
Scaffold-GS	0.993	0.979	0.981	0.995	0.985	0.992	0.982	0.971	0.984
GS-Shader	0.992	0.964	0.979	0.994	0.985	0.990	0.980	0.966	0.981
Ours-w anchor	0.994	0.979	0.982	0.994	0.993	0.992	0.984	0.975	0.987
Ours	0.995	0.982	0.984	0.995	0.994	0.994	0.985	0.978	0.988

Table 9. **Per-scene SSIM comparison on the NSVF dataset.**

	Bike	Life	Palace	Robot	Space	Steam	Toad	Wine	Avg.
TensoRF	0.010	0.048	0.022	0.010	0.020	0.017	0.031	0.051	0.026
Tri-MipRF	0.012	0.048	0.023	0.019	0.019	-	0.036	0.055	0.030
NeuRBF	0.006	0.036	0.016	0.009	0.011	0.011	0.025	0.036	0.019
3D-GS	0.005	0.028	0.017	0.006	0.009	0.007	0.018	0.025	0.015
Scaffold-GS	0.007	0.030	0.019	0.008	0.019	0.010	0.022	0.021	0.017
GS-Shader	0.007	0.051	0.020	0.008	0.016	0.010	0.023	0.029	0.020
Ours-w anchor	0.005	0.027	0.018	0.007	0.007	0.008	0.021	0.025	0.015
Ours	0.004	0.023	0.015	0.005	0.007	0.007	0.017	0.021	0.013

Table 10. **Per-scene LPIPS (VGG) comparison on the NSVF dataset.**

	Teapot	Plane	Record	Ashtray	Dishes	Headphone	Jupyter	Lock	Avg.
3D-GS	27.24	26.80	43.81	34.43	29.62	38.72	40.52	29.36	33.81
Scaffold-GS	30.64	29.14	47.79	35.66	32.12	37.19	40.04	30.13	35.34
Ours-w anchor	33.53	31.56	50.35	36.14	32.95	38.48	40.10	30.96	36.76
Ours	34.88	30.83	50.51	37.02	32.90	39.45	41.18	31.46	37.28

Table 11. **Per-scene PSNR comparison on our "Anisotropic Synthetic" dataset.**

	Teapot	Plane	Record	Ashtray	Dishes	Headphone	Jupyter	Lock	Avg.
3D-GS	0.968	0.946	0.994	0.969	0.947	0.989	0.985	0.932	0.966
Scaffold-GS	0.979	0.965	0.998	0.973	0.967	0.986	0.983	0.924	0.972
Ours-w anchor	0.985	0.973	0.999	0.974	0.973	0.988	0.984	0.930	0.976
Ours	0.987	0.967	0.998	0.977	0.967	0.990	0.986	0.943	0.977

Table 12. **Per-scene SSIM comparison on our "Anisotropic Synthetic" dataset.**

	Teapot	Plane	Record	Ashtray	Dishes	Headphone	Jupyter	Lock	Avg.
3D-GS	0.043	0.085	0.019	0.044	0.120	0.015	0.075	0.098	0.062
Scaffold-GS	0.029	0.057	0.006	0.038	0.082	0.021	0.086	0.099	0.052
Ours-w anchor	0.022	0.042	0.004	0.039	0.067	0.017	0.084	0.093	0.046
Ours	0.021	0.051	0.009	0.036	0.087	0.014	0.071	0.087	0.047

Table 13. **Per-scene LPIPS (VGG) comparison on our "Anisotropic Synthetic" dataset.**

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Plenoxels	21.91	20.10	23.49	20.66	22.25	27.59	23.62	23.42	24.67
iNGP	22.17	20.65	25.07	23.47	22.37	29.69	26.69	29.48	30.69
Mip-NeRF360	24.37	21.73	26.98	26.40	22.87	31.63	29.55	32.23	33.46
3D-GS	25.08	21.41	27.26	26.62	22.68	31.54	29.04	31.44	32.16
Scaffold-GS	25.05	21.20	27.33	26.49	23.23	32.13	29.44	31.59	32.49
Ours-w/o anchor	25.11	21.31	27.48	26.59	22.63	31.84	30.05	31.91	33.38
Ours	25.12	21.63	27.50	26.61	23.19	32.14	30.11	32.10	33.68

Table 14. **Per-scene PSNR comparison on the Mip-NeRF 360 dataset.**

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Plenoxels	0.496	0.431	0.606	0.523	0.509	0.842	0.759	0.648	0.814
iNGP	0.512	0.486	0.701	0.594	0.542	0.871	0.817	0.858	0.906
Mip-NeRF360	0.685	0.583	0.813	0.744	0.632	0.913	0.894	0.920	0.941
3D-GS	0.746	0.588	0.855	0.769	0.635	0.924	0.913	0.931	0.944
Scaffold-GS	0.738	0.568	0.846	0.754	0.641	0.927	0.914	0.929	0.946
Ours-w/o anchor	0.739	0.584	0.856	0.759	0.631	0.925	0.919	0.933	0.946
Ours	0.744	0.589	0.850	0.758	0.640	0.929	0.917	0.930	0.950

Table 15. **SSIM Comparison on the Mip-NeRF 360 dataset.**

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Plenoxels	0.506	0.521	0.386	0.503	0.540	0.419	0.441	0.447	0.398
iNGP	0.446	0.441	0.257	0.421	0.450	0.261	0.306	0.195	0.205
Mip-NeRF360	0.301	0.344	0.170	0.261	0.339	0.211	0.204	0.127	0.176
3D-GS	0.245	0.359	0.123	0.242	0.347	0.199	0.184	0.117	0.182
Scaffold-GS	0.266	0.383	0.143	0.276	0.353	0.200	0.195	0.121	0.186
Ours-w/o anchor	0.247	0.361	0.121	0.246	0.349	0.203	0.180	0.115	0.184
Ours	0.251	0.346	0.137	0.258	0.341	0.192	0.184	0.120	0.174

Table 16. **LPIPS Comparison on the Mip-NeRF 360 dataset.**

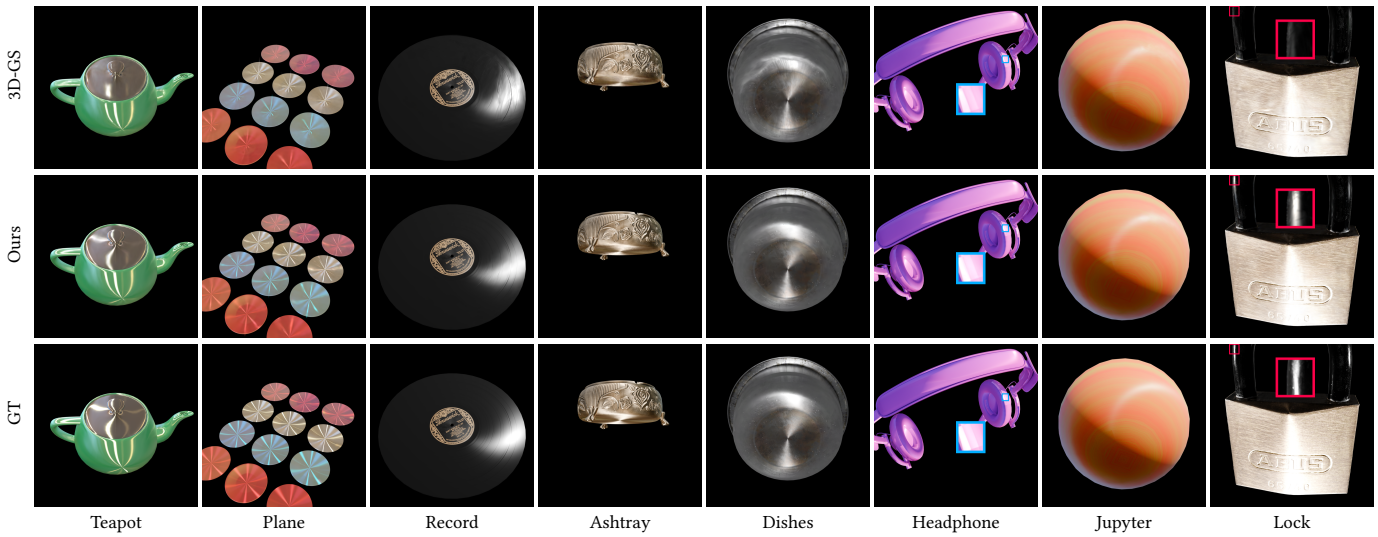

Fig. 11. **Visualization on our "Anisotropic Synthetic" dataset.** We show the comparison between our method and 3D-GS across all eight scenes. Qualitative experimental results demonstrate the significant advantage of our method in modeling anisotropic scenes, thereby enhancing the rendering quality of 3D-GS.

Method	Truck			Train			Dr Johnson			Playroom		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM ↑	LPIPS↓
Plenoxels	23.22	0.774	0.335	18.93	0.663	0.422	23.14	0.787	0.521	22.98	0.802	0.499
iNGP	23.38	0.800	0.249	20.46	0.689	0.360	28.26	0.854	0.352	21.67	0.779	0.428
Mip-NeRF360	24.91	0.857	0.159	19.52	0.660	0.354	29.14	0.901	0.237	29.66	0.900	0.252
3D-GS	25.42	0.878	0.147	22.01	0.811	0.209	29.21	0.900	0.247	30.09	0.898	0.247
Scaffold-GS	25.77	0.883	0.147	22.15	0.822	0.206	29.80	0.907	0.250	30.62	0.904	0.258
Ours-w/o anchor	25.50	0.878	0.150	22.38	0.813	0.211	29.24	0.902	0.252	30.17	0.900	0.246
Ours	26.25	0.885	0.144	22.90	0.825	0.204	29.89	0.906	0.251	31.00	0.906	0.253

Table 17. **Quantitative comparison on Tanks&Temples and Deep Blending dataset.**