Normal-GS: 3D Gaussian Splatting with Normal-Involved Rendering

Meng Wei[1] **Qianyi Wu**[1] **Jianmin Zheng**[2] **Hamid Rezatofighi**[1] **Jianfei Cai**[1]
[1]Monash Univeristy [2]Nanyang Technological University
`{meng.wei,qianyi.wu,hamid.rezatofighi,jianfei.cai}@monash.edu`
`{ASJMZheng}@ntu.edu.sg`

Abstract

Rendering and reconstruction are long-standing topics in computer vision and graphics. Achieving both high rendering quality and accurate geometry is a challenge. Recent advancements in 3D Gaussian Splatting (3DGS) have enabled high-fidelity novel view synthesis at real-time speeds. However, the noisy and discrete nature of 3D Gaussian primitives hinders accurate surface estimation. Previous attempts to regularize 3D Gaussian normals often degrade rendering quality due to the fundamental disconnect between normal vectors and the rendering pipeline in 3DGS-based methods. Therefore, we introduce **Normal-GS**, a novel approach that integrates normal vectors into the 3DGS rendering pipeline. The core idea is to model the interaction between normals and incident lighting using the physically-based rendering equation. Our approach re-parameterizes surface colors as the product of normals and a designed Integrated Directional Illumination Vector (IDIV). To save memory usage and simplify the optimization, we employ an anchor-based 3DGS to implicitly encode locally-shared IDIVs. Additionally, Normal-GS leverages optimized normals and Integrated Directional Encoding (IDE) to accurately model specular effects, enhancing both rendering quality and surface normal precision. Extensive experiments demonstrate that Normal-GS achieves near state-of-the-art visual quality while obtaining accurate surface normals and preserving real-time rendering performance.

1 Introduction

Radiance Fields have emerged as a prominent representation in 3D vision, largely propelled by the pioneering advancements of Neural Radiance Fields (NeRF) [1]. Despite their success, NeRFs are hindered by prolonged rendering times and cumbersome training processes. Recently, 3D Gaussian Splatting (3DGS) has been introduced to represent Radiance Fields using millions of 3D Gaussians, each endowed with additional attributes such as opacity and color [2]. 3DGS significantly enhances rendering efficiency through CUDA-accelerated rasterization [3], achieving comparable or superior fidelity to NeRF-based models. The advantages of 3DGS have garnered significant attention [4], prompting numerous studies aiming at enhancing its rendering capabilities [5–7] and improving the quality of its underlying geometry [8–10].

However, a critical issue continues to plague the development of 3DGS: *the quality of appearance and geometry seemingly oscillates like a seesaw.* Efforts to refine the geometry often involve the incorporation of regularization techniques [8, 11] or additional geometry supervision [12, 13], which may compromise rendering fidelity. Conversely, advancements in appearance modeling, such as the integration of implicit networks [6] and sophisticated shading attributes [5, 7], struggle to concurrently elevate both appearance and geometry quality. This raises a pertinent question: *Is it feasible to achieve a high-quality appearance while also capturing precise underlying geometry information in 3DGS?*

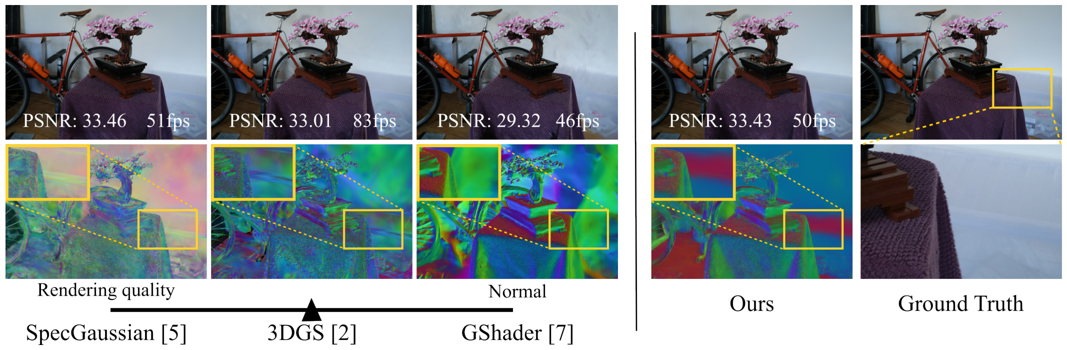


Figure 1: The "seesaw" characteristic between the rendering quality and the normal accuracy of 3DGS-based methods. Our **Normal-GS** is able to efficiently achieve accurate normal estimation while preserving competitive rendering quality. Our method successfully captures the normals of the cover of the semi-transparent box behind the table.

Revisiting the classical rendering equation [14], we identify a fundamental issue in the appearance-geometry conflict: the disconnection between surface normals and the rendering process in current 3DGS methods. Specifically, the existing methods render pixel colors by alpha-blending colored 3D Gaussians. The color of each Gaussian is queried by the function value among Spherical Harmonics [15] along the ray direction, which disregards the contributions of surface normals. This oversight hampers the ability to achieve a balanced integration of appearance and geometry. Our framework draws inspiration from the conventional rendering equation, which incorporates surface normals into appearance calculations. Although prior attempts have explored this integration in contexts such as inverse rendering [16, 17] and specular appearance modeling [7, 5], they leverage simplified model [18] or approximation approaches like split-sum [19] or global environmental map [20, 21] to decompose appearance into several individual components with meticulous regularization terms, which either compromising in rendering quality or geometric accuracy. This challenge highlights the complexity of balancing various components for physically-based rendering, prompting our design of a straightforward method to integrate normal information explicitly and simply into Gaussian appearance modeling.

In this work, we propose a normal-involved shading for 3DGS, called **Normal-GS**, for high-quality rendering and accurate normal estimation, while maintaining real-time performance. Our method directly accounts for normal contributions in the 3DGS rendering pipeline, enabling gradient signals to backpropagate properly to better align 3D Gaussians. Specifically, Normal-GS considers the normal and incident lighting interactions at the surface from the physically-based shading perspective and explicitly models normals' contributions for diffuse and specular components. By considering the low-frequency nature of the incident light field, we further implicitly model it with MLPs. With our strategy, compared with the state-of-the-art (SOTA) 3DGS methods, we achieve competitive rendering quality, more accurate normal information, and real-time rendering, see one example in Fig. 1. Our contributions are summarized below.

- We propose a new formulation to involve normal information in appearance modeling for 3DGS, which strives for a good balance between appearance and geometry modeling.
- We leverage an effective structure to regularize our framework using anchor-based MLPs, without introducing many extra regularization terms and achieving better quality.
- Extensive experiments demonstrate our framework for obtaining superior view synthesis and normal estimation. Our design could serve as a plug-in component for 3DGS approaches.

2 Related Works

Radiance Fields: From Neural Radiance Fields to 3D Gaussian Splatting. Neural Radiance Field (NeRF) has significantly influenced the field of 3D vision with its impressive novel view synthesis capabilities [1]. NeRF catalyzed advancements in neural rendering, inspiring a multitude of subsequent research focused on enhancing rendering quality under complex lighting and material conditions [22–24], varying camera distributions [25–27], and scalability for large scenes [28–30].

However, the implicit representation used in NeRF, which relies on a Multi-layer Perceptron (MLP) to compute density and radiance for any given 3D position and ray direction, necessitates extensive computations. This inefficiency has spurred efforts to enhance the practicality of NeRF, with notable strides made in accelerating its processing efficiency [2, 31, 32].

Among these advancements, 3D Gaussian Splatting (3DGS) has emerged as a promising solution [2]. By explicitly modeling 3D scenes using sets of 3D Gaussians and employing tile-based GPU rasterization, 3DGS achieves real-time rendering with competitive quality. Its success has spurred applications in areas like 3D generation [33, 34], physical simulation [35], and sparse view reconstruction [36, 37], despite its higher memory requirements for scene representation. Innovations such as feature anchoring in Scaffold-GS [6], value pruning and quantization techniques [38–40] and mining spatial relationship [41–43] are among the efforts to reduce the memory footprint of 3DGS, thereby enhancing storage efficiency and rendering fidelity.

Geometry and Appearance in 3DGS: From Surface Reconstruction to Inverse Rendering. Geometry and appearance are central to 3D reconstruction, and the discrete and explicit nature of 3DGS can result in noisy underlying geometry [8, 10, 11, 13, 44]. SuGaR [8] was an initial attempt to refine mesh extraction from 3DGS, applying regularization to align the Gaussian Splatting with the actual surface contours. To better define geometry, capturing normal information is essential, leading to innovations like transforming 3DGS into 2D Gaussian Splatting (2DGS) and Gaussian Surfels [10, 44]. These methods apply regularization to depth and normal rendering, ensuring that the Gaussians are appropriately distributed across surfaces. While these approaches improve surface reconstruction, they often sacrifice rendering fidelity in novel view synthesis due to the lack of a clear relationship between geometry and appearance [7, 10, 44].

In appearance modeling, the focus has shifted to inverse rendering for 3DGS, which aims to separate scene elements into materials, lighting, and geometries. This inherently unconstrained problem challenges traditional rendering equations. Techniques such as Monte Carlo integration, point-based ray tracing [17, 45], and baking [16] are employed to handle the complex integrals, often relying on simplified models like the Disney Bidirectional Reflectance Distribution Function (BRDF) [18] and approximation methods such as split-sum [16, 17, 45, 46] or environmental mapping [7, 46, 47]. Despite their efforts, these simplifications generally result in lower rendering quality, as reflected in reduced Peak Signal-to-Noise Ratios (PSNR) [16, 17, 45], failing to match the original 3DGS.

It is worth noting that some concurrent works [47, 48] have a similar motivation as ours. [48] simultaneously enhance geometry accuracy and rendering quality by incorporating a laborious dual-branch framework, which uses 3DGS for appearance rendering and an implicit neural surface for geometry production. While this approach integrates the strengths of each system, it is hampered by slower training speeds relative to standalone 3DGS, due to its additional networks for surface representation. [47] also distills geometry information from an extra neural implicit surface network for deferred rendering. Our solution proposes a new perspective from a rendering standpoint to consider geometry and appearance simultaneously without an extra implicit network for the surface.

3 Method

Our goal is to simultaneously enhance both image quality and normal estimation capability of 3DGS, while maintaining real-time rendering. To achieve this, we first analyze the existing rendering pipeline in 3DGS to identify its limitations in not involving surface normals in color modeling (Sec. 3.1). We then design a normal-involved rendering strategy that aligns with the physically based rendering equation. Our method effectively models interactions between normals and incident lighting, parameterized by the proposed Integrated Directional Illumination Vectors (IDIV) (Sec. 3.2). We employ an anchor-based GS to implicitly encode locally shared IDIVs to save memory and aid optimization (Sec. 3.3). The training details of our framework are provided in Sec. 3.4.

3.1 Preliminary

3D Gaussian Splatting [2]. 3DGS models scenes using a set of discrete 3D Gaussians, each defined by its spatial mean $\boldsymbol{\mu}$ and covariance matrix Σ:

$$G(\boldsymbol{p}) = \exp(-\frac{1}{2}(\boldsymbol{p}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{p}-\boldsymbol{\mu})). \tag{1}$$

The covariance matrix Σ is parameterized by a scaling matrix S and a rotation matrix R, such that $\Sigma = RSS^T R^T$, ensuring it remains positive semi-definite during optimization.

Each 3D Gaussian is associated with a color c and an opacity α. During rendering, these Gaussians are projected (rasterized) onto the image plane, forming 2D Gaussian splats $G'(x)$, as described in [3]. The 2D Gaussian splats are sorted from front to back tile-wisely, and α-blending [49] is performed for each pixel x to render its color as follows:

$$C(x) = \sum_{i \in N} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \sigma_j), \quad \sigma_i = \alpha_i G'_i(x) \tag{2}$$

where N specifies the number of 2D Gaussian splats covering the current pixel. Heuristic densification and pruning strategies [2] are employed to address potential under- and over-reconstruction and ensure multi-view consistency in rendered images.

The color of each Gaussian, c, is represented by Spherical Harmonics (SH) as $k_l^m Y_l^m(\boldsymbol{\omega}_{\text{view}})$ to provide view-dependent effects, where (l, m) is the degree and order of the SH basis Y_l^m, k_l^m is the corresponding SH coefficient, and $\boldsymbol{\omega}_{\text{view}}$ specifies the viewing direction. 3DGS usually uses a maximum degree l of 3, formulating $c(\boldsymbol{\omega}_{\text{view}}) = \sum_{l=0}^{3} \sum_{m=-l}^{l} k_l^m Y_l^m(\boldsymbol{\omega}_{\text{view}})$.

In Eq. (2), standard 3DGS treats colors as intrinsic attributes, independent of surface normals. This independence prevents surface normals from receiving gradient signals during the backpropagation pass when optimizing surface colors. Moreover, this separation significantly undermines our goal of simultaneously enhancing image quality and normal estimation capability, since improvements in normals cannot contribute to the rendering quality in the forward shading pass of 3DGS.

3.2 Normal-Involved Shading Strategy

Physically Based Rendering. To integrate surface normals into both the backward and forward rendering passes of 3DGS effectively, we propose a normal-involved rendering strategy. This strategy adopts principles from physically based surface rendering [14] in computer graphics, which models the out radiance L_{out} of a surface point as a function of the incident lighting L_{in} and normals. Specifically, for each 3D point, its out-radiance in the outward direction $\boldsymbol{\omega_o} = -\boldsymbol{\omega}_{\text{view}}$ is defined as:

$$c(\boldsymbol{\omega}_{\text{view}}) = L_{\text{out}}(\boldsymbol{\omega_o}) = L_{\text{E}}(\boldsymbol{\omega_o}) + \int_{\Omega^+} L_{\text{in}}(\boldsymbol{\omega_i})(\boldsymbol{\omega_i} \cdot \boldsymbol{n}) f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})\, d\boldsymbol{\omega_i} \tag{3}$$

where L_{E} is the emitted radiance along the outward direction, the integral is defined over the upper hemisphere Ω^+ of the surface; $\boldsymbol{\omega_i}$ is the incident light direction; BRDF $f_r(\cdot)$ describes how light is reflected from $\boldsymbol{\omega_i}$ to $\boldsymbol{\omega_o}$; the cosine term $(\boldsymbol{\omega_i} \cdot \boldsymbol{n})$ accounts for the light geometric attenuation.

Integrated directional illumination vectors for Lambertian objects. We first consider a simple ideal case where the scene contains only Lambertian objects. For more complex materials, we handle them in Sec. 3.4. Given that the diffuse reflection of Lambertian objects is view-independent, the complicated BRDF function $f_r(\cdot)$ can be reduced to a spatially varying albedo term k_{D}. Omitting the emitted radiance L_{E}, the rendering equation accounting for the diffuse reflectance L_{D} becomes:

$$L_{\text{D}} = \int_{\Omega^+} L_{\text{in}}(\boldsymbol{\omega_i}) \cdot k_{\text{D}} \cdot (\boldsymbol{\omega_i} \cdot \boldsymbol{n})\, d\boldsymbol{\omega_i} = k_{\text{D}} \int_{\Omega^+} L_{\text{in}}(\boldsymbol{\omega_i})(\boldsymbol{\omega_i} \cdot \boldsymbol{n})\, d\boldsymbol{\omega_i}. \tag{4}$$

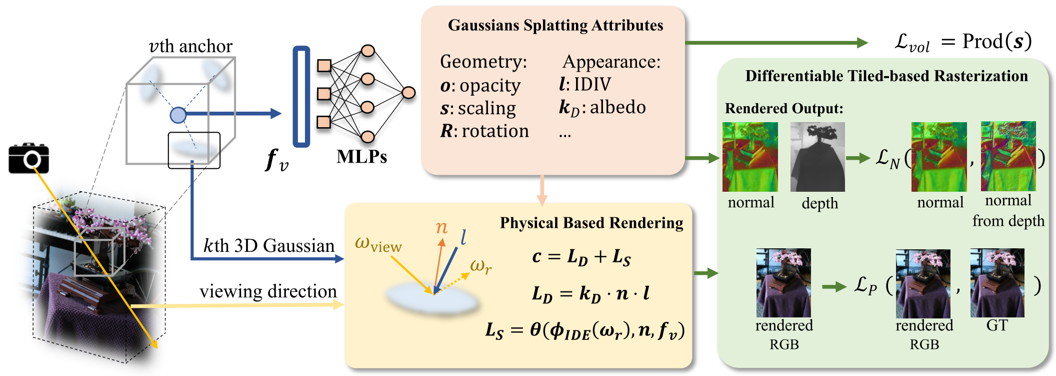
We want to find a way to explicitly re-parameterize L_{D} as a function of normal $\boldsymbol{n}$, meanwhile avoiding complicated integrals to save time. Specifically, we want to represent L_{D} as a dot product between $\boldsymbol{n}$ and some illumination vector $\boldsymbol{l}$. Inspired by the traditional shape-from-shading strategy [50], we extract the normal out of Eq. (4):

$$L_{\text{D}} = k_{\text{D}} \cdot \boldsymbol{n} \cdot \left[\int_{\Omega^+} L_{\text{in}}(\boldsymbol{\omega_i}) \boldsymbol{\omega_i}\, d\boldsymbol{\omega_i} \right]. \tag{5}$$

By defining a new term called "*integrated directional illumination vector* (IDIV)" as

$$\boldsymbol{l} = \int_{\Omega^+} L_{\text{in}}(\boldsymbol{\omega_i}) \boldsymbol{\omega_i}\, d\boldsymbol{\omega_i}, \tag{6}$$

we model the color for Lambertian objects as the albedo times a dot product between IDIV and the surface normal: $c = L_{\text{D}} = k_{\text{D}} \cdot \boldsymbol{n} \cdot \boldsymbol{l}$. Compared with the original 3DGS and the method [7] that directly use a constant diffuse color, our re-parameterization successfully accounts for the normal's contributions to the diffuse component of surface colors.



vth anchor
f_v
MLPs
Gaussians Splatting Attributes
Geometry:
o: opacity
s: scaling
R: rotation
Appearance:
l: IDIV
k_D: albedo
...
$\mathcal{L}_{vol} = \text{Prod}(s)$
Differentiable Tiled-based Rasterization
Rendered Output:
normal
depth
$\mathcal{L}_N$ (normal , normal from depth)
kth 3D Gaussian
viewing direction
ω_{view}
n
l
ω_r
Physical Based Rendering
$c = L_D + L_S$
$L_D = k_D \cdot n \cdot l$
$L_S = \theta(\phi_{IDE}(\omega_r), n, f_v)$
rendered RGB
$\mathcal{L}_P$ (rendered RGB , GT)

Figure 2: Our normal-involved GS, **Normal-GS**, reparameterizes the original colors into the diffuse and specular components, $c = L_\text{D} + L_S$ (bottom). It models the diffuse component as the dot product between the normal vector $\boldsymbol{n}$ and the Integrated Directional Illumination Vector (IDIV) $\boldsymbol{l}$, and utilizes the Integrated Directional Encoding (IDE) [22] to capture view-dependent specular effects. Inherent parameters are encoded implicitly by a locally shared anchor Gaussian (left) and decoded using MLPs (top). Our method accounts for the contributions of normals to colors, effectively enhancing geometry accuracy and rendering quality.

Analysis. Our re-parameterization of the diffuse color enables the normal vector to be computationally involved in the rendering pass. As such, during backpropagation, by the chain rule, the surface color could pass gradient signals to the normal vectors as follows

$$\frac{d\mathcal{L}}{d\boldsymbol{n}} = \frac{d\mathcal{L}}{dc} \cdot \frac{dc}{d\boldsymbol{n}} = \frac{d\mathcal{L}}{dc} \cdot (k_\text{D} \cdot \boldsymbol{l}). \tag{7}$$

Other image-based methods [7, 16, 46] usually optimize a single environment map E and query this map using normals to determine diffuse colors. Consequently, their gradients on normals function by rotating the 3D Gaussians until the queried environment map color matches the pixel color, making the optimization process heavily dependent on the quality of E. These methods assume the existence of a global environment map, which may not be valid for general real scenes. In contrast, our approach employs IDIVs to capture local incident lighting, eliminating the reliance on such a strong assumption and providing a more flexible and accurate framework for optimizing surface normals.

3.3 Modeling Locally Shared Integrated Directional Illumination Vectors

Replacing 3D Gaussian colors c with $k_\text{D} \cdot \boldsymbol{n} \cdot \boldsymbol{l}$ introduces additional free parameters. Therefore, it is necessary to regularize the solution space of the introduced variables k_D and $\boldsymbol{l}$ to stabilize the training process and avoid overfitting.

Regularizing the solution space. Xu et al. [50] address solution space constraints from an optimization perspective. They have introduced several regularizers, such as total variation and Laplacian terms, to enhance the performance. However, their mesh-based method relies on a connected topology, which is incompatible with the discrete and sparse nature of 3D Gaussians. Although k-NN can be used to find connected primitives, it severely hampers training speed. Moreover, since 3D Gaussians can be semi-transparent, directly applying their methods to our case is unsuitable.

Instead, we approach the problem from a neural rendering perspective. As observed in [6, 42, 51, 52], 3D Gaussians with similar textures usually cluster together. A similar situation applies to incident lighting, which tends to be locally shared and exhibits low-frequency variations. To avoid the redundancy of storing Integrated Directional Illumination Vectors (IDIVs) for each Gaussian and capture local coherence, we propose to encode IDIVs with locally shared features and decode them using MLPs. An additional advantage of using MLPs to approximate IDIVs is their inherent smoothness in function approximations, aligning with our objectives.

In particular, we employ an anchor-based Gaussian Splatting (GS) method, Scaffold-GS [6], to implicitly represent IDIVs using locally shared features $\boldsymbol{f}_v$ stored at anchors v and decode them using a global MLP, $\theta_l(\boldsymbol{f}_v) = \{\boldsymbol{l}_v^k\}_{k=1}^K$, where K IDIVs share a local feature $\boldsymbol{f}_v$. Therefore, we only need to learn a compact set of per-anchor local features and use the MLP to generate per-Gaussian IDIVs, significantly reducing the dimensionality of the problem. Further details on our model configurations can be found in the following Sec. 3.4 and Appendix A.2.

3.4 Training Details

Our method adopts the established 3DGS pipeline [2], which first performs Structure-from-Motion [53] on input images to generate sparse points. These points serve as initial 3D Gaussians and are optimized to model the scene.

Defining surface normals on 3D Gaussians. During optimization, 3D Gaussians often exhibit flatness around the surface areas, as observed in [5, 7, 45]. For a near-flat Gaussian ellipsoid, its shortest axis thus functions as the normal vector. However, the shortest axis of 3D Gaussians does not physically represent the actual surface normal. In contrast, implicit radiance fields leverage the gradient of densities to naturally approximate surfaces, and the Signed Distance Function (SDF) has its gradients as surface normals by definition. To make the shortest axis of 3D Gaussians align with the real surface normal, we employ a depth regularized loss term on normals, similar to [7, 10, 44].

We first render the depth and normal images, $\{\mathcal{D}, \mathcal{N}\}$, using the 3DGS tile-based rasterizer, which is achieved by replacing the color term in Eq. 2 with depths and normals of 3D Gaussians, respectively. Then we compute the image-space gradients of the depth image $\nabla_{(u,v)}\mathcal{D}$ and finally perform the cross product of the gradients, resulting in $\mathcal{N}_\mathcal{D}$. The depth regularized loss term on normals is defined as $\mathcal{L}_N = 1 - \mathcal{N}_\mathcal{D} \cdot \mathcal{N}$. Although there are other methods for improving normals, since our primary goal is to introduce a normal-involved shading method, we find this self-regularizer is enough for our models. Other methods are welcome to be combined with our strategy for future improvements.

Capturing specular effects. Our derivation on Sec. 3.2 is based on the Lambertian assumption. To make our framework more generalizable, similar to our parameterization of the diffuse term, we aim to express the specular term, L_s, as a function of the normal vector $\boldsymbol{n}$. Unlike view-independent diffuse colors, specular reflectance is highly sensitive to the view direction due to the complex BRDF, $f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$. Since normal vectors are implicitly involved in the BRDF to capture effects such as the Fresnel effect [18], we cannot directly extract the normal vector as we did for IDIVs in Sec. 3.2. Inspired by the Ref-NeRF [22], we model the specular component as a function of the reflection direction of the viewing direction with surface normals involved: $\boldsymbol{\omega_r} = 2(\boldsymbol{\omega_o} \cdot \boldsymbol{n})\boldsymbol{n} - \boldsymbol{\omega_o}$. By applying the Integrated Directional Encoding (IDE) [22] on the reflection direction, we have $L_\text{S} = \theta(\phi_\text{IDE}(\boldsymbol{\omega}_\text{r}), \boldsymbol{n}, \boldsymbol{f}_v)$. The final color of the Gaussian is the summation of L_D and L_S. More details can be found in the Appendix.

Model architecture. Figure. 2 illustrates the whole process of our method. We follow [6] to define a set of anchor Gaussians $\{v\}$, where each v is associated with a position $\boldsymbol{x}_v$, a local feature $\boldsymbol{f}_v$, a scaling factor $\boldsymbol{s}_v$ and k offsets $\boldsymbol{O}_v^k$ for k nearby 3D Gaussians. Anchor-wise features together with global MLPs are employed to predict attributes of 3D Gaussians. We utilize our normal-involved rendering to compute surface colors $c = L_\text{D} + L_\text{S}$. Colors are finally gathered following the traditional 3DGS pipeline, together with rendered normals and depths for self-regularization. The final loss $\mathcal{L}$ consists of the original photo-metric loss $\mathcal{L}_\text{P}$ used in 3DGS [2], the volume regularization loss $\mathcal{L}_\text{vol}$ used in Scaffold-GS [6] and the self-regularized depth-normal loss $\mathcal{L}_\mathcal{N}$:

$$\mathcal{L} = \mathcal{L}_\text{P} + \lambda_\text{vol}\mathcal{L}_\text{vol} + \lambda_\mathcal{N}\mathcal{L}_\mathcal{N}. \tag{8}$$

4 Experiments

We evaluated our Normal-GS method against several state-of-the-art 3DGS-based methods across multiple datasets, presenting both qualitative and quantitative results.

Baselines. We selected the following baseline methods for comparison: 1) *3DGS* [2] and *Scaffold-GS* [6]: These served as the vanilla and baseline methods. 2) *GaussianShader (GShader)* [7] and *SpecGaussian* [5]: These 3DGS-based methods modeled specular effects and used surface normals. Both methods computed the reflected direction $\boldsymbol{\omega}_\text{r}$ of the viewing direction w.r.t the normal. GShader

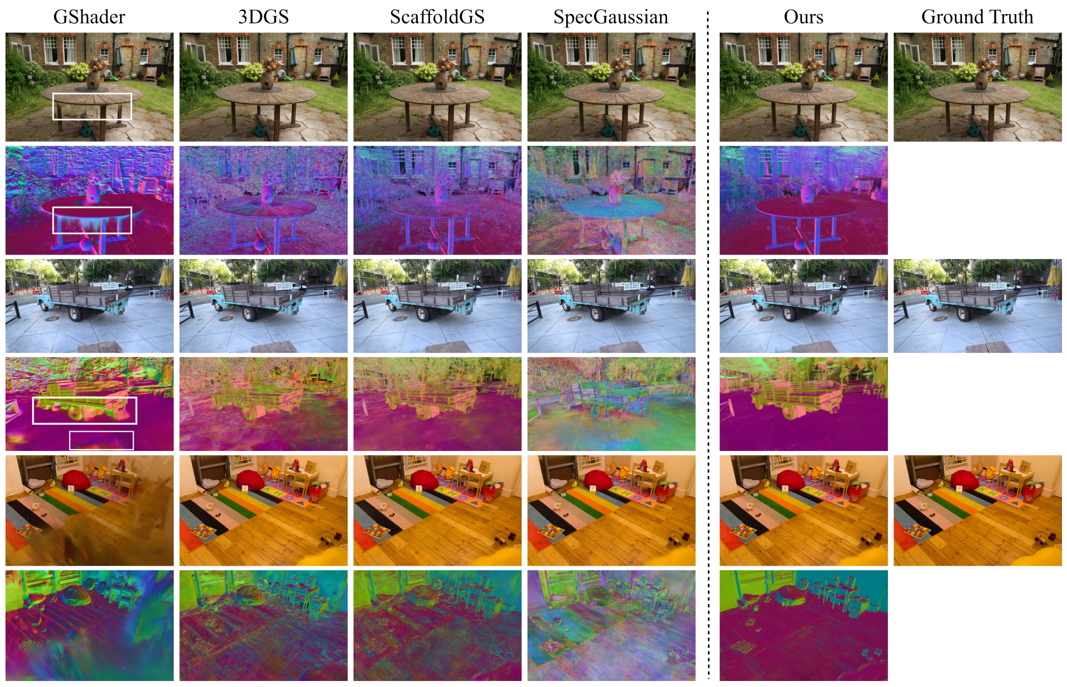


Figure 3: **Qualitative comparisons of the rendering quality and normal estimation.** Our method produced clean normals estimation and preserved good rendering quality.

used $\boldsymbol{\omega}_{\mathrm{r}}$ to query a learned environment map for shading, while SpecGaussian used $\boldsymbol{\omega}_{\mathrm{r}}$ to query the learned Spherical Gaussians. Additionally, we designed a new baseline method based on Scaffold-GS. In the original Scaffold-GS, anchor features $\boldsymbol{f}_v$ were passed into a color MLP to predict 3D Gaussian colors. In our new baseline method, we also fed normals into the same color MLP, implicitly incorporating normal information. We referred to this baseline method as 3) *ScaffoldGS w/ N*.

Datasets and evaluation metrics. We followed the original 3DGS [2] methodology and used the NeRF Synthetic [1], Mip-NeRF 360 [26], Tank and Temple [54], and Deep Blending [55] datasets to demonstrate the performance of our method. Due to licensing issues, we tested on 7 out of 9 scenes in the Mip-NeRF 360 dataset. Following [2, 26], we selected every 8th image for testing and used the remaining images for training. We reported PSNR, SSIM, and LPIPS to measure rendering quality and the Mean Angular Error (MAE) of normals on the NeRF Synthetic [1] dataset.

Implementation details. We implemented our method in Python using the PyTorch framework [56]. Our rasterization pipeline followed the CUDA-based rasterizer described in [2]. We trained our models for 30k iterations, following the settings of baseline methods. Consistent with [2, 6], we set $\lambda_{\text{vol}} = 0.001$. For the depth-normal loss, we used $\lambda_{\mathcal{N}} = 0.01$. Because the depth and normals were inaccurate at the start of training, we added the depth-normal loss after training for 5k iterations. We will release our code after publication. More implementation details were included in the Appendix A.2.

We tested our method and the baseline methods using their original released implementations with default hyperparameters on an NVIDIA RTX 3090 GPU with 24 GB of memory. To obtain normals for comparisons, we retrained most of the baseline methods, except for 3DGS, which provided its optimized data. For methods without defined normals, such as 3DGS [2] and Scaffold-GS [6], we used the shortest axis of 3D Gaussians as normals for evaluation, consistent with ours.

4.1 Results

We first present a comprehensive comparison of rendering quality and normal estimation in Fig.3, demonstrating that our method either matches or surpasses state-of-the-art (SOTA) approaches in metrics of PSNR/SSIM/LPIPS (Tab.1). Notably, our approach significantly outperforms the baseline

Table 1: **Quantitative comparisons of the rendering quality.** Our methods achieve comparable or even better results compared with the SOTA approach, SpecGaussian [5]. However, SpecGaussian performs much worse in normal estimation, while ours achieves a good balance between geometry and appearance quality. † denotes the results quoted from their paper. GShader* fails on the Deep Blending scene.

	Mip-NeRF360			Tanks & Temples			Deep Blending		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS†	28.691	0.870	0.182	23.142	0.840	0.183	29.405	0.903	0.243
ScaffoldGS	29.267	0.869	0.190	24.088	0.851	0.175	30.140	0.905	0.256
ScaffoldGS w/ N	29.177	0.870	0.192	23.976	0.852	0.176	30.163	0.905	0.263
GShader	26.060	0.825	0.233	21.262	0.785	0.254	19.159*	0.769*	0.453*
SpecGaussian	29.287	0.864	0.196	24.502	0.855	0.175	30.114	0.905	0.252
Normal-GS (Ours)	29.341	0.869	0.194	24.219	0.854	0.174	30.187	0.910	0.252

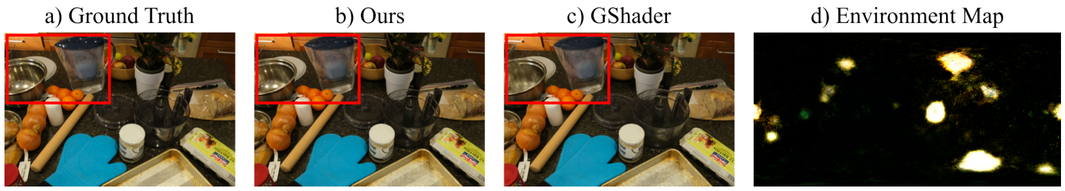


Figure 4: b) Our method successfully captures specular effects. c) GShader [7] relying on an environment map performs poorly when d) its learned environment map fails. Scenes are taken from the Mip-NeRF 360 dataset [26], which have complicated lighting conditions.

method (ScaffoldGS) and its variant incorporating normals (ScaffoldGS w/ N) in almost all metrics. This comparison underscores the limitations of the implicit normal utilization in ScaffoldGS w/N, which fails to accurately model appearance. In contrast, our method adopts an explicit modeling of color as a function of normals, directly derived from the rendering equation. This explicit integration enhances both the forward and backward passes during training, providing clearer cues for the network to capture nuanced color details, thereby improving rendering quality.

Our approach also demonstrates superior rendering performance compared to methods like GShader [7], which depend on global environment maps to model specular effects, as evidenced in Tab. 1. The dependency of normal gradients on environment map quality, described by the equation $\frac{dL}{dN} = \frac{dL}{dE}\frac{dE}{dN}$ [7, 47], often leads to inaccuracies in capturing detailed information, as shown in Fig.4 and further evidenced by performance issues on challenging datasets like DeepBlending[55] in Tab. 1. Our framework's capability to learn local incident light field attributes tied to distinct anchors in 3D space allows for more effective handling of complex lighting effects.

Although SpecGaussian [5] achieves competitive rendering quality through the use of Spherical Gaussian encoding and neural networks, it falls short in accurate normal modeling, as depicted in Fig.3. This suggests the insufficiency of relying solely on neural networks for integrating normal information into geometric modeling. Our method, with its explicit integration of normals into the rendering process, not only maintains high rendering quality but also ensures the accuracy and smoothness of normals. Our approach underscores the crucial role of direct geometric involvement in the rendering pipeline for achieving both high geometric accuracy and visual fidelity. More results are available in Appendix A.3.

To quantitatively evaluate normal estimation, we conducted tests on the Synthetic-NeRF dataset, which includes ground-truth normal data. As shown in Tab. 2(a), our method outperforms others in the normal Mean Angular Error (MAE) metric. Visual results in Tab. 2(b) confirm our method's superior capability in capturing accurate geometry alongside high-fidelity rendering, as highlighted in the yellow box, showcasing the effectiveness of our design.

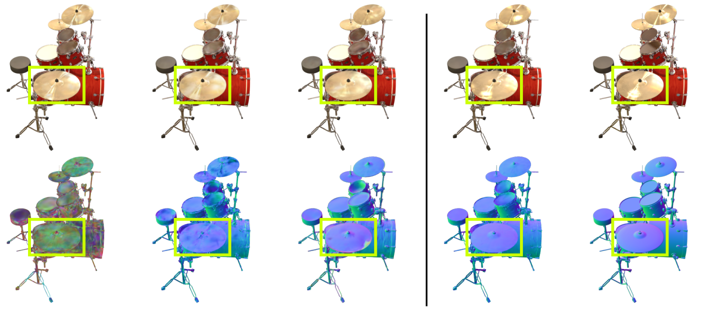
4.2 Ablation studies

Our contributions are significant in two key areas: the integration of Integrated Directional Illumination Vectors (IDIV) and the implementation of an anchor-based regularization design. We

Table 2: **Comparison on Synthetic-NeRF dataset.** (a) The numerical results on normal consistency. Our method achieved state-of-the-art results among all methods. (b) We visualized the rendered normals for qualitative comparison. Our framework produced the best geometry.

	Normals MAE ↓
SpecGaussian [5]	45.98
ScaffoldGS [6]	25.56
GShader [7]	23.56
Normal-GS (Ours)	20.71

(a) Normals comparision



(b) Visualization of rendered normals

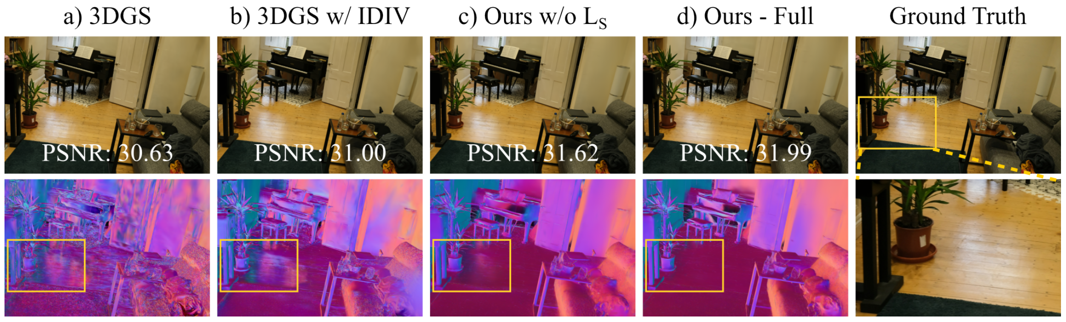


Figure 5: **Ablation study about our proposed components.** The IDIV together with our regularization strategy produces better quality in rendering quality and normal accuracy.

initially validate the impact of IDIV through an ablation study conducted within a conventional 3DGS framework, detailed in Sec. 3.2. Introducing IDIV necessitates additional parameters, requiring the adoption of regularization techniques such as Laplacian and Total-Variation loss, referenced in [50]. The outcomes, illustrated in Fig. 5 (a) and (b), demonstrate substantial enhancements in rendering quality and the smoothing of normals.

However, we found that the model's performance is highly sensitive to the tuning of loss weights. To address this, we implemented an anchor-based IDIV regularization strategy in Sec. 3.3 with results illustrated in Fig. 5 (c). This approach employs neural networks to model locally shared IDIVs, proving to be both robust and efficient in regularizing the solution space. Notably, the simplicity yet effectiveness of our final loss term, defined in Eq. 8, underscores the potential of our design to enhance both geometric and rendering quality. When considering the specular component, our full (Fig. 5 (d)) model achieves superior results.

Table 3: Quantitative ablation studies on DTU.

Model	mCD ↓	PSNR ↑
(a): Scaffold-GS [6]	1.84	38.14
(b): (a) + $\mathcal{L}_{\mathcal{N}}$	0.95	35.90
(c): (b) + $L_{specular}$	0.93	37.37
(d): Full model	0.94	37.63

We also performed quantitative ablation studies on the DTU dataset [57] to further validate the effectiveness of our components. The DTU dataset contains 15 scenes, each with scanned geometries provided as ground truth. We conducted the ablation study by comparing the mean Chamfer distance (mCD) for geometry accuracy and PSNR for rendering quality evaluation. We followed [44] to pre-process the data, extract meshes and evaluate obtained results. As shown in Tab. 3, our base model is Scaffold-GS [6]. We add the self-regularized depth-normal loss, $\mathcal{L}_{\mathcal{N}}$, as defined in Sec.3.4. The results of (b) in Tab. 3 indicate that the depth-normal loss can significantly improve the geometry quality but ruin the rendering with more than 2dB PSNR drop. Then we introduce the specular component into (b), which can improve the rendering quality

Table 4: Rendering quality comparisons with NeRF-based methods on the Mip-NeRF 360 dataset.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Mip-NeRF 360 [26]	29.231	0.844	0.207
Ref-NeRF [22]	28.553	0.849	0.196
Zip-NeRF [58]	30.077	0.876	0.170
Ours	29.341	0.869	0.194

Table 5: Geometric and rendering quality comparisons on the DTU dataset.

Method	mCD ↓	PSNR ↑
3DGS [2]	1.96	35.76
SuGaR [8]	1.33	34.57
2DGS [44]	0.80	34.52
Ours	0.94	37.63

without compromising the geometry, which further assures the importance of involving normal into the rendering. Finally, we add the proposed IDIV to get our final model (d), which further improves the PSNR by more than 0.2 dB with similar geometry quality. Compared with the base model (a), our full model achieves a better balance between the geometry quality and the rendering fidelity.

4.3 Geometric reconstruction comparisons

In addition to evaluating normal accuracy, we compare the overall geometry quality of our method with existing 3DGS-based approaches [2, 8, 44] on the DTU dataset [57]. For a fair comparison, we follow the pipeline described in [44] and measure the mean Chamfer distance (mCD) and PSNR to assess both geometric reconstruction and rendering quality. The results for 3DGS [2], SuGaR [8], and 2DGS [44] are directly adopted from [44]. As Tab. 5 illustrates, our method achieves outstanding rendering quality while maintaining better geometric quality than 3DGS and SuGaR. 2DGS, however, sacrifices some rendering fidelity in novel view synthesis due to the lack of a clear relationship between geometry and appearance, as seen in Tab. 5. Unlike 2DGS, our method explicitly addresses the interactions between lighting and normals. It is also worth noting that 2DGS and our approach are conceptually orthogonal and could potentially be combined for further improvements.

4.4 Comparisons with NeRF-based methods

Finally, for completeness, we conduct comparisons with existing state-of-the-art NeRF-based methods, including Mip-NeRF 360 [26], Ref-NeRF [22], and Zip-NeRF [58]. In Tab. 4, we present their original results for 7 out of the 9 scenes from the Mip-NeRF 360 dataset [26], aligning with our experimental setup. Our method performs the second best in terms of the PSNR and achieves comparable SSIM and LPIPS scores. However, the training time for Zip-NeRF exceeds 10 hours on high-performance GPUS. In contrast, 3DGS-based methods, including ours, are relatively fast and can support real-time rendering, offering a significant speed advantage over NeRF-based approaches.

5 Discussions

Conclusion. We introduce a novel shading technique within the 3D Gaussian Splatting (3DGS) framework to improve view synthesis and normal estimation. By integrating Directional Illumination Vectors (IDIV) as advanced attributes for color representation and utilizing an anchor-based design with a neural MLP for IDIV prediction, our approach achieves a superior balance between rendering quality and geometric accuracy. This enhancement not only elevates image fidelity but also ensures precise normal estimations, pushing the boundaries of 3DGS technology in real-world applications.

Limitation and Future Work. We acknowledge existing limitations within our framework. Notably, the self-regularized normal loss proves suboptimal for certain outdoor scenes at a distance (as observed in the Appendix 7), negatively impacting rendering quality in these regions. This issue could potentially be addressed with more sophisticated normal guidance techniques. In future research, we plan to delve into a more granular decomposition of the Integrated Directional Illumination Vectors (IDIV) and specular components. This exploration aims to enhance capabilities in texture editing and relighting, thereby broadening the practical applications of our framework.

Boarder Impact. As our approaches require per-scene optimization to obtain the 3D model for each scene, the computational resources for model training could be a concern for global climate change.

References

[1] 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. *Communications of the ACM*, 65 (1):99–106, 2021.

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, August 2023. ISSN 0730-0301, 1557-7368. doi: 10.1145/3592433.

[3] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001. doi: 10.1109/VISUAL.2001.964490.

[4] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *arXiv preprint arXiv:2403.11134*, 2024.

[5] Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. *arXiv preprint arXiv:2402.15870*, 2024.

[6] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[7] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[8] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[9] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. *arXiv preprint arXiv:2402.14650*, 2024.

[10] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. doi: 10.1145/3641519.3657441.

[11] Hanlin Chen, Chen Li, and Gim Hee Lee. Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023.

[12] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[13] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing. *arXiv preprint arXiv:2403.17822*, 2024.

[14] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[15] Ernest William Hobson. *The theory of spherical and ellipsoidal harmonics*. CUP Archive, 1931.

[16] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[17] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[18] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *Acm Siggraph*, volume 2012, pages 1–7. vol. 2012, 2012.

[19] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 4(3):1, 2013.

[20] Ned Greene. Environment mapping and other applications of world projections. *IEEE computer graphics and Applications*, 6(11):21–29, 1986.

[21] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, 2001.

[22] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022.

[23] Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Neilf: Neural incident light field for physically-based material estimation. In *European Conference on Computer Vision*, pages 700–716. Springer, 2022.

[24] Jingyang Zhang, Yao Yao, Shiwei Li, Jingbo Liu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Neilf++: Inter-reflectable light fields for geometry and material estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3601–3610, 2023.

[25] 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.

[26] 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.

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

[28] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12922–12931, 2022.

[29] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4150–4159, 2023.

[30] 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.

[31] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021.

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

[33] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations*, 2023.

[34] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *CVPR*, 2024.

[35] Tianyi Xie, Zeshun Zong, Yuxin Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *CVPR*, 2024.

[36] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[37] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024.

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

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

[40] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. *arXiv preprint arXiv:2401.02436*, 2023.

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

[42] Yihang Chen, Qianyi Wu, Jianfei Cai, Mehrtash Harandi, and Weiyao Lin. Hac: Hash-grid assisted context for 3d gaussian splatting compression. *arXiv preprint arXiv:2403.14530*, 2024.

[43] Runyi Yang, Zhenxin Zhu, Zhou Jiang, Baijun Ye, Xiaoxue Chen, Yifei Zhang, Yuantao Chen, Jian Zhao, and Hao Zhao. Spectrally pruned gaussian fields with neural compensation. *arXiv preprint arXiv:2405.00676*, 2024.

[44] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. doi: 10.1145/3641519.3657428.

[45] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jingtuo Liu, Liangjun Zhang, Jian Zhang, Bin Zhou, et al. Gir: 3d gaussian inverse rendering for relightable scene factorization. *arXiv preprint arXiv:2312.05133*, 2023.

[46] Keyang Ye, Qiming Hou, and Kun Zhou. 3d gaussian splatting with deferred reflection. *arXiv preprint arXiv:2404.18454*, 2024.

[47] Tong Wu, Jia-Mu Sun, Yu-Kun Lai, Yuewen Ma, Leif Kobbelt, and Lin Gao. Deferredgs: Decoupled and editable gaussian splatting with deferred shading. *arXiv preprint arXiv:2404.09412*, 2024.

[48] Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. Gsdf: 3dgs meets sdf for improved rendering and reconstruction. *arXiv preprint arXiv:2403.16964*, 2024.

[49] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 40(4), June 2021. URL `http://www-sop.inria.fr/reves/Basilic/2021/KPLD21`.

[50] Di Xu, Qi Duan, Jianmin Zheng, Juyong Zhang, Jianfei Cai, and Tat-Jen Cham. Shading-based surface detail recovery under general unknown illumination. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):423–436, 2017.

[51] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv preprint arXiv:2312.00732*, 2023.

[52] Evangelos Ververas, Rolandos Alexandros Potamias, Jifei Song, Jiankang Deng, and Stefanos Zafeiriou. Sags: Structure-aware 3d gaussian splatting. *arXiv preprint arXiv:2404.19149*, 2024.

[53] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[55] 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 (Proc. SIGGRAPH Asia)*, 37(6):257:1–257:15, 2018.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[57] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014.

[58] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023.

A Appendix

A.1 Modeling the specular component using Integrated Directional Encodings (IDE)

As discussed in Sec. 3.4, we utilize the Integrated Directional Encoding (IDE) of the Ref-NeRF [22] to model the specular component. Specifically, the IDE encodes the distribution of the reflection vectors using a set of spherical harmonics $\{Y_l^m\}$. This distribution follows the von Mises-Fisher (vMF) distribution, with the reflection vector ω_r as the center and the inverse roughness $\kappa = 1/\rho$ as the concentration parameter. κ describes the width of the vMF distributions, corresponding to the surface roughness. With this vMF distribution $\hat{\omega} \sim \text{vMF}(\hat{\omega}_r, \kappa)$, Verbin et al. [22] define the IDE as

$$\text{IDE}(\hat{\omega}_r, \kappa) = \{\mathbb{E}_{\hat{\omega}\sim\text{vMF}(\hat{\omega}_r,\kappa)}[Y_l^m(\hat{\omega})] : (l, m) \in \mathcal{M}_L\} \tag{9}$$

with $\mathcal{M}_L = \{(l, m) : l = 1, ..., 2^L, m = 0, ..., l\}$. Verbin et al. [22] further show that the above expectation equals $A_l(\kappa)Y_l^m(\hat{\omega}_r)$ with

$$A_l(\kappa) \approx \exp{(-\frac{l(l+1)}{2\kappa})}. \tag{10}$$

For a detailed derivation and illustration, we recommend referring to their original paper [22].

After obtaining the IDE of the reflection direction, we pass $\phi_{\text{IDE}}(\boldsymbol{\omega}_r)$, $\boldsymbol{n}$ and the adjacent anchor feature $\boldsymbol{f}_v$ as inputs to a global MLP θ, generating the final specular color as $L_S = \theta(\phi_{\text{IDE}}(\boldsymbol{\omega_r}), \boldsymbol{n}, \boldsymbol{f}_v)$. Details about our MLP are provided as follows.

A.2 Implementation details

Neural network architectures We employ an anchor-based GS method, Scaffold-GS [6], to predict attributes of 3D Gaussians. The overall architecture closely follows its original implementation, with modifications made to the color modeling components. Specifically, we introduce two additional components to predict diffuse L_D and specular colors L_S respectively. Other components including the updating strategy remain unchanged. The neural network architecture of the newly introduced MLPs is consistent with those used in [6]. These MLPs are fully connected networks with one hidden layer and utilize the ReLU activation function. The learning rates and feature dimensions for these MLPs are set to match the original color MLP used by Scaffold-GS.

Defining normals on 3D Gaussians Recall that the covariance matrix Σ of a 3D Gaussian is parameterized by a scaling matrix S and a rotation matrix R, $\Sigma = RSS^TR^T$, formulating a 3D ellipsoid. Let $\Lambda = SS^T = \text{diag}(s_1^2, s_2^2, s_3^3)$, the eigenvalues of Λ are thus (s_1^2, s_2^2, s_3^3), with the dummy eigenvectors $(\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3})$. In $\Sigma = RSS^TR^T = R\Lambda R^T$, the rotation matrix $R = [\boldsymbol{r_1}; \boldsymbol{r_2}; \boldsymbol{r_3}]$ acts like rotating eigenvectors of Λ and preserves its eigenvalues. Consequently, the eigenvectors of the covariance matrix Σ indicate the principal axes of the Gaussian ellipsoid and the eigenvalues correspond to the axes' lengths. The shortest axis of the Gaussian ellipsoid thus corresponds to the ith column of the rotation matrix, where $s_i = \min(s_1, s_2, s_3)$.

Loss Our loss is defined as $\mathcal{L} = \mathcal{L}_{\text{P}} + \lambda_{\text{vol}}\mathcal{L}_{\text{vol}} + \lambda_{\mathcal{N}}\mathcal{L}_{\mathcal{N}}$, with $\lambda_{\mathcal{N}} = 0.01$ and $\lambda_{\text{vol}} = 0.001$. The photometric loss, as defined in [2], is $\mathcal{L}_{\text{p}} = (1 - \lambda_{\text{SSIM}})\mathcal{L}_1 + \lambda_{\text{SSIM}}\mathcal{L}_{\text{D-SSIM}}$, with $\lambda_{\text{SSIM}} = 0.2$. The volumetric loss, as defined in [6], is $\mathcal{L}_{\text{vol}} = \sum_{i=1}^{N_g} \text{Prod}(s_i)$, where $\text{Prod}(\cdot)$ stands for the product of the values of a vector and s_i is the scales of each 3D Gaussian. Our self-regularized depth-normal loss $\mathcal{L}_{\mathcal{N}}$ has been discussed in Sec. 3.4.

A.3 More results

We include per-scene results, following the same evaluation strategies and metrics discussed in Sec. 4. We also provide additional qualitative visualization results of baseline methods and ours in Fig. 6.

Table 6: Per-scene PSNR ↑ for the Mip-NeRF360 [26] dataset.

Method	bicycle	garden	stump	room	counter	kitchen	bonsai	Avg.
3DGS	25.25	27.41	26.55	30.63	28.70	30.32	31.98	28.691
ScaffoldGS*	25.18	27.51	26.56	31.92	29.56	31.53	32.61	29.267
ScaffoldGS w/ N	25.20	27.47	26.56	31.84	29.47	31.41	32.29	29.177
GShader*	22.74	26.44	24.66	24.82	26.60	27.68	29.48	26.060
SpecGaussian*	24.96	27.43	26.22	31.80	29.60	31.68	33.32	29.287
Ours	25.23	27.64	26.54	31.99	29.71	31.61	32.67	29.341

Limitations Our method uses the self-regularized depth-normal loss, which deteriorates the performance of normal estimation and rendering at distant regions. Figure. 7 illustrates one failure case of our methods.

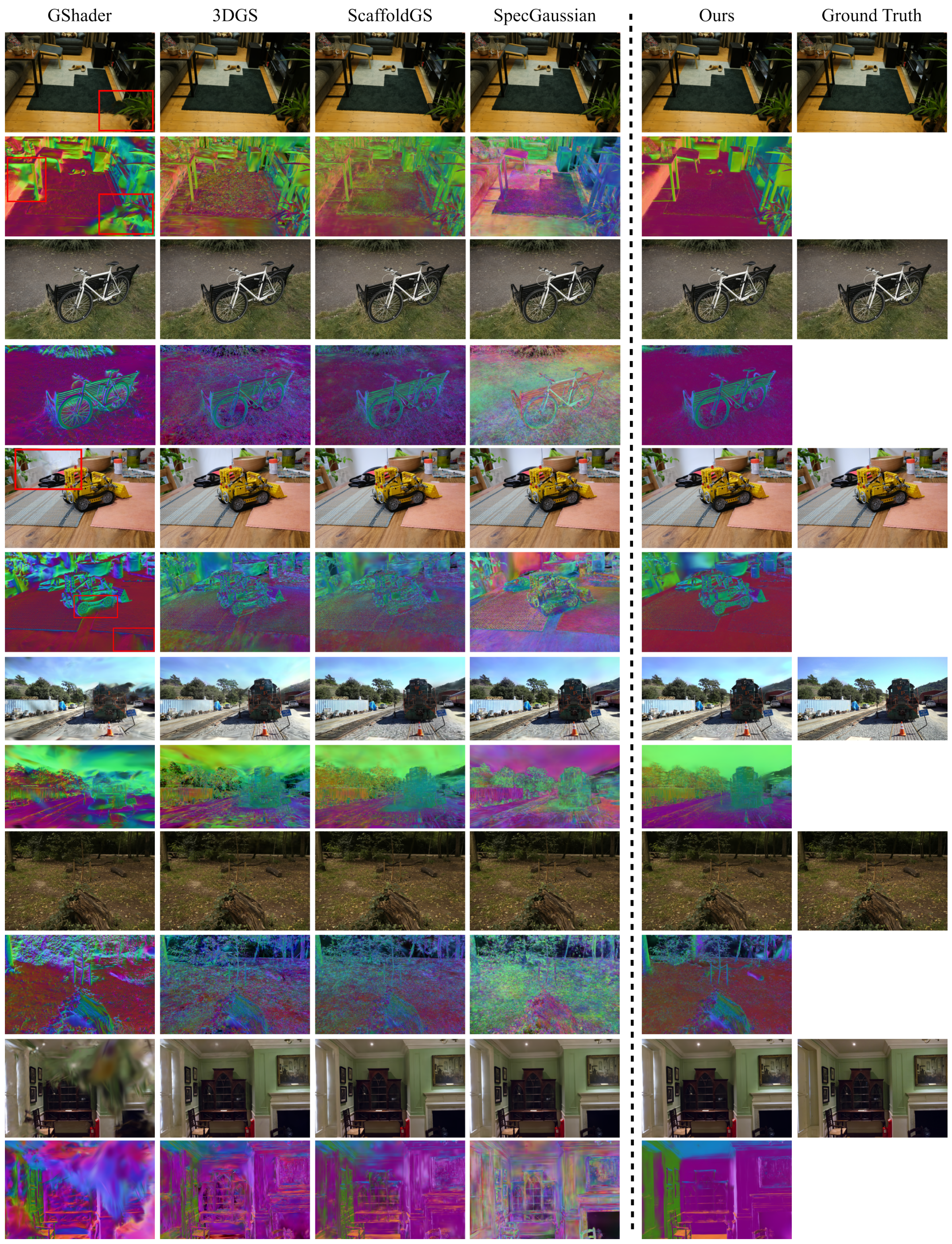


Figure 6: **More qualitative comparisons of the rendering quality and normal estimation.** Our method produced clean normals estimation and preserved good rendering quality.

A.4 Metric definition.

Besides the common-used image quality metric in novel view synthesis evaluation in the main context, we provide a geometry metric evaluation about normal MAE scores for qualitative comparison. Here we provide the definition of it.

Denote the rendering normal as $\mathcal{N}$ and the ground-truth 2D normal map $\mathcal{N}_{GT}$, we aim to evaluate the similarity between these two images. Assume there is a mask M for object definition under each testing view, we use it to

Table 7: Per-scene SSIM ↑ for the Mip-NeRF360 [26] dataset.

Method	bicycle	garden	stump	room	counter	kitchen	bonsai	Avg.
3DGS	0.771	0.868	0.775	0.914	0.905	0.922	0.938	0.870
ScaffoldGS*	0.758	0.863	0.766	0.922	0.912	0.926	0.941	0.869
ScaffoldGS w/ N	0.759	0.862	0.767	0.923	0.912	0.927	0.943	0.870
GShader	0.692	0.843	0.702	0.846	0.875	0.897	0.917	0.825
SpecGaussian*	0.753	0.852	0.754	0.918	0.904	0.924	0.943	0.864
Ours	0.757	0.863	0.764	0.922	0.911	0.926	0.943	0.869

Table 8: Per-scene LPIPS ↓ for the Mip-NeRF360 [26] dataset.

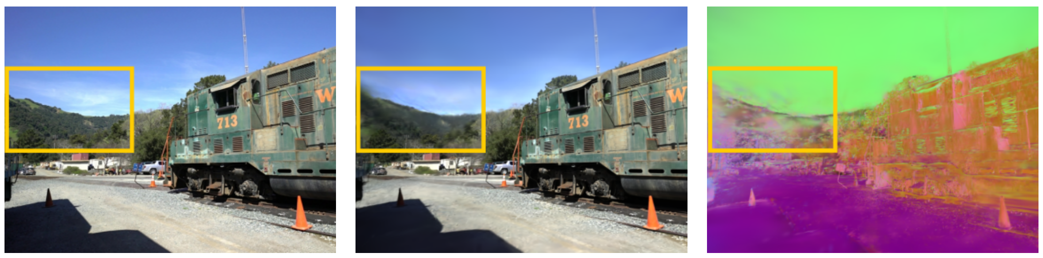
Method	bicycle	garden	stump	room	counter	kitchen	bonsai	Avg.
3DGS	0.205	0.103	0.210	0.220	0.204	0.129	0.205	0.182
ScaffoldGS*	0.232	0.118	0.236	0.213	0.198	0.128	0.204	0.190
ScaffoldGS w/ N	0.236	0.121	0.241	0.213	0.201	0.128	0.206	0.192
GShader	0.278	0.130	0.277	0.305	0.243	0.160	0.242	0.233
SpecGaussian*	0.235	0.131	0.242	0.220	0.211	0.131	0.200	0.196
Ours	0.243	0.120	0.244	0.216	0.202	0.129	0.209	0.194

Table 9: Quantitative comparisons of the rendering quality on the Tanks and Temples dataset [54].

	Truck			Train		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS	25.187	0.879	0.148	21.097	0.802	0.218
ScaffoldGS*	25.825	0.881	0.145	22.350	0.820	0.206
ScaffoldGS w/ N	25.738	0.882	0.147	22.214	0.821	0.205
GShader*	23.696	0.843	0.193	18.828	0.727	0.315
SpecGaussian*	26.112	0.885	0.145	22.893	0.825	0.205
Ours	25.858	0.882	0.145	22.580	0.826	0.203

Table 10: Quantitative comparisons of the rendering quality on the Deep Blending dataset [55].

	Playroom			Dr. Johnson		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS	30.044	0.906	0.241	28.766	0.899	0.244
ScaffoldGS*	30.655	0.906	0.256	29.624	0.904	0.257
ScaffoldGS w/ N	30.785	0.907	0.263	29.541	0.903	0.263
GShader*	21.007	0.807	0.406	17.312	0.730	0.499
SpecGaussian*	30.589	0.905	0.252	29.639	0.904	0.252
Ours	30.846	0.913	0.260	29.529	0.906	0.257



(a) Ground Truth (b) Rendered (c) Normal

Figure 7: The failure case of our method. Our method performs suboptimally for scenes at a distance.

mask out the rendering normal and ground-truth normal by $M(\mathcal{N})$ and $M(\mathcal{N}_{GT})$, we then calculate the average angle between the normals vectors for all valid pixels, *i.e.*, Normal MAE= $\frac{\sum_{i \in M} \arccos < M_i(\mathcal{N}), M_i(\mathcal{N}_{GT}) >}{\sum_{i \in M} M_i}$

B Assets License

In Table 11, we list the licenses of all the existing assets including the code and data we have used in this work.

Asset	License Link
3D Gaussian Splatting [2]	https://github.com/graphdeco-inria/gaussian-splatting/blob/main/LICENSE.md
Scaffold-GS [6]	https://github.com/city-super/Scaffold-GS/blob/main/LICENSE.md
GaussianShader [7]	https://github.com/Asparagus15/GaussianShader/blob/main/LICENSE.md
SpecGaussian [5]	https://github.com/ingra14m/Specular-Gaussians/blob/main/LICENSE
COLMAP [53]	https://colmap.github.io/license.html
Synthesic-NeRF [1]	https://drive.google.com/drive/folders/128yBriW1IG_3NJ5Rp7APSTZsJqdJdfc1 (CC-BY 3.0)
MipNeRF360 [26]	https://jonbarron.info/mipnerf360/
DeepBlending [55]	http://visual.cs.ucl.ac.uk/pubs/deepblending/datasets.html
Tank and Tamples [54]	https://www.tanksandtemples.org/license/ (CC-BY 4.0)

Table 11: **Licenses of existing assets we have used in this work.**

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction demonstrate the paper's contribution to the scope of 3D Gaussian Splatting, considering the issue of appearance-geometry conflict and proposing a solution for it.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This paper discusses the limitations in the Discussion Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: In the theory derivation in Sec. 3.2, we point out the assumption of Lambertian surface for proposed components.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The information to reproduce the main experimental results is provided in Sec. 4 and appendix, including network configuration, and hyper-parameters for training.

   Guidelines:

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

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

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

   Answer: [No]

   Justification: The details for implementation are provided in the appendix and we will release our code upon publication.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The training and testing details are elaborated in the appendix and Sec. 4.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [No]

   Justification: We report the per-scene results in the appendix, which can be used to calculate statistical significance.

   Guidelines:

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provide the related information in the experiments and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research is carefully conducted with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The discussion about potential negative social impacts is mentioned in Sec. 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper aims to support novel view synthesis and geometric reconstruction, which poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We include the assets used in our paper (code and dataset) in Sec. B.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA]

    Justification: This paper does not release new assets during the time of submission.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

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

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.