SpecTRe-GS: Modeling Highly Specular Surfaces with Reflected Nearby Objects by Tracing Rays in 3D Gaussian Splatting

Jiajun Tang[1,2] Fan Fei[1,2] Zhihao Li[3] Xiao Tang[3] Shiyong Liu[3]
Youyu Chen[4] Binxiao Huang[5] Zhenyu Chen[3] Xiaofei Wu[3] Boxin Shi[1,2#]
[1]State Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University
[2]National Engineering Research Center of Visual Technology, School of Computer Science, Peking University
[3]Huawei Noah's Ark Lab [4]Harbin Institute of Technology [5]University of Hong Kong

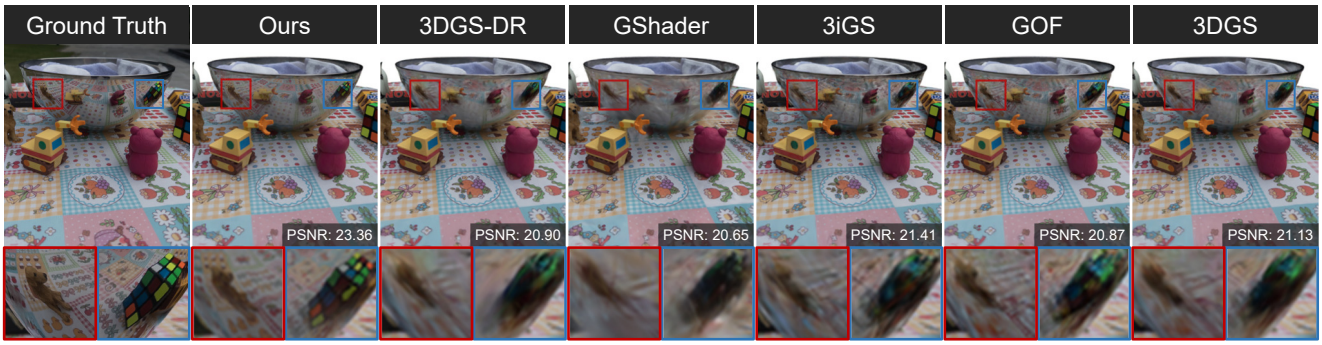

Figure 1. Our SpecTRe-GS effectively captures the appearance of reflectors with highly specular surfaces, even in the presence of nearby objects. In the real-world scene shown above, it performs favorably against previous methods [13, 15, 30, 39, 42], both visually (as the highlighted reflections within the insets) and quantitatively (with PSNR values within the reflective region annotated in the corners).

Abstract

*3D Gaussian Splatting (3DGS), a recently emerged multi-view 3D reconstruction technique, has shown significant advantages in real-time rendering and explicit editing. However, 3DGS encounters challenges in the accurate modeling of both high-frequency view-dependent appearances and global illumination effects, including inter-reflection. This paper introduces **SpecTRe-GS**, which addresses these challenges and models highly **Spec**ular surfaces that reflect nearby objects through **T**racing **R**ays in 3D **G**aussian **S**platting. SpecTRe-GS separately models reflections from highly specular and rough surfaces to leverage the distinctions between their reflective properties and integrates an efficient ray tracer within the 3DGS framework for querying secondary rays, thus achieving fast and accurate rendering. Also, it incorporates normal prior guidance and joint geometry optimization at various stages of the training process to enhance geometry reconstruction for undistorted reflections. Experiments on both synthetic and real-world scenes demonstrate the superiority of SpecTRe-GS compared to existing 3DGS-based methods in capturing highly specular inter-reflections and also showcase its editing applications.*

1. Introduction

Recently, 3D Gaussian Splatting (3DGS) [15] has emerged as an innovative paradigm for multi-view 3D reconstruction. Unlike its predecessor, neural radiance fields (NeRF) [23], 3DGS adopts an explicit point-based scene representation with tile-based rasterization rather than relying on ray casting through an implicit neural field for rendering. This transition endows 3DGS with significant advantages in terms of real-time rendering speed and allows for convenient post-reconstruction editing. Nevertheless, challenges arise when the scene to be reconstructed includes highly specular (*i.e.*, mirror-like) surfaces which exhibit strong and high-frequency view-dependent appearances. The original 3DGS struggles to accurately model such high-frequency reflections, as it utilizes per-Gaussian third-order spherical harmonics (SH) to represent view-dependent appearances. This approach is restricted to low-frequency radiance variations in angular space and is susceptible to overfitting when the training views that observe the Gaussian are not sufficiently dense [17, 37].

To address this issue, a natural strategy is to adopt a physically-based shading model and optimize reflective properties for each Gaussian [2, 7, 8, 10, 11, 13, 18, 29, 35, 36, 39, 40, 43, 47], rather than merely recording the out-

The work was done during an internship in Huawei.
#Corresponding author. E-mail: shiboxin@pku.edu.cn.

going radiance. This strategy enables the computation of reflected radiance by evaluating the physically constrained rendering equation [14], taking into account local scene components including geometry, material properties, and incident lighting, thereby yielding more accurate, realistic, and reliable view-dependent effects. Among the three local scene components, addressing local incident lighting is perhaps the most intricate. Adopting a suboptimal approach to addressing local incident lighting may undermine global illumination effects such as inter-reflections, resulting in an unrealistic scene appearance. Notably, this issue is particularly pronounced in scenes characterized by highly specular surfaces; for instance, the absence of reflections of nearby objects would be especially noticeable. Therefore, it is crucial for methods aimed at modeling such high-frequency view-dependent appearances through the rendering equation to carefully address local incident lighting.

Unfortunately, effectively acquiring local incident lighting poses a substantial challenge for 3DGS-based methods. The advantages of 3DGS come with a trade-off: The exclusive reliance on the tile-based rasterizer limits its ability to efficiently render secondary rays emanating from vastly different origins towards arbitrary directions, which is essential for querying local incident lighting. As a compromise, a majority of methods neglect secondary effects [36, 39, 47] or periodically bake secondary effects including local occlusions [8, 40] and incident radiance [18, 29] as low-order SH stored in Gaussians or probes. However, low-order SH can only capture low-frequency variations in incident lighting, and the requirement of baking not only incurs additional storage costs but also hinders post-reconstruction editing, such as inserting or deleting nearby objects or relighting. Employing screen space ray tracing (SSRT) [2] produces high-frequency but incomplete inter-reflections as SSRT struggles to trace out-of-screen and occluded rays.

We propose **SpecTRe-GS** to model highly **Spec**ular surfaces with reflected nearby objects through **T**racing **R**ays in the 3D space within the **G**aussian **S**platting framework, overcoming the aforementioned challenges. We first made the observation that, although highly specular surfaces exhibit reflections that can be extremely high-frequency and thus hard to capture using SH color or neural network, they are also favorable for modeling in the sense that the specular lobe of their bidirectional reflectance distribution functions (BRDF) are highly concentrated. This fact facilitates the evaluation of the rendering equation [14] on these surfaces because tracing only a few secondary rays already suffices, in contrast to rougher surfaces with more scattered specular lobes. To make use of this fact, we separately model highly specular and rough surfaces, ray-tracing the appearance of the former while retaining the low-order SH color to well-approximate the latter, significantly reducing the amount of secondary rays to trace. To further speed up the time-consuming ray tracing process, we implement a ray tracer within the 3DGS framework, inspired by the recent progresses [24, 42]. We also propose a training strategy to enhance the scene geometry reconstruction, the quality of which directly decides the quality of the reconstructed reflection. To summarize, our contributions are as follows:

- proposing a shading model tailored for indirect mirror reflections combined with an efficient ray tracer for fast and accurate rendering;
- designing a training strategy to enhance geometry reconstruction for precise incident direction computing; and
- achieving state-of-the-art novel view synthesis results on scenes with highly specular reflections (shown in Fig. 1).

2. Related Work

Improving view-dependent appearances in 3DGS. The original 3DGS [15] uses low-order SH color that captures only low-frequency angular variations, a limitation that many follow-up methods were trying to address (see Tab. 1). While a few methods [30, 35] (row 2) adopt neural networks to decode local features, the majority [2, 8, 10, 13, 18, 29, 36, 39, 40, 47] (rows 3-8) employ the physically-based rendering equation [14]. This equation offers the most accurate approach for modeling view-dependent appearances but requires explicit modeling of surface normals, BRDFs, and the most demanding component – incident lighting. The rendering equation can accommodate all-frequency BRDFs; thus, the frequency of the reflected radiance is now bounded by that of the modeled incident lighting. While an environment map suffices to model direct lighting, it is significantly more challenging to model indirect lighting responsible for global illumination effects including occlusions and inter-reflections. Some methods circumvent this issue by ignoring secondary effects [36] (row 3) or by using a residual color [13] (row 4). For methods that explicitly model indirect lighting, a common strategy is to periodically bake occlusions [8] or inter-reflections [18, 29] into low-order SH, thereby limiting their incident radiance to low-frequency variations (rows 5-7). GI-GS [2] (row 8) uses screen space ray tracing to simulate high-frequency inter-reflections; however, it only models those whose sources are directly visible in the viewport. There are also specialized methods for plane mirrors that reflect the point cloud or camera about the plane [19, 22]. Our SpecTRe-GS traces the incident lighting in 3D space, thus is able to render high-frequency and complete inter-reflections for highly specular surfaces with convex shapes.

Ray-based 3D Gaussian rendering. It is noteworthy that tracing secondary rays for incoming local lighting has almost become a common practice in recent NeRF-based methods that explicitly model specular reflections or inter-reflections [9, 20, 32, 33, 45]. However, it is challenging to achieve this in 3DGS-based frameworks. 3DGRT [24]

Table 1. We summarize works improving the view-dependent appearances upon 3DGS [15] based on: How do they model the view-dependent appearances (*i.e.*, the outgoing radiance) and the supported frequency in angular space; how do they accommodate the global illumination effects (*i.e.*, the local incident lighting) including occlusions and inter-reflections and the supported frequency in angular space; and whether is it feasible to edit the scene appearance under (in)direct lighting after reconstruction without any re-baking.

Method	View-dependent Appearance		Global Illumination Effect			Editable Appearance
	Modeling	Frequency	Occlusion	Inter-reflection	Frequency	
(1) 3DGS [15]	Low-order SH	Low	Jointly baked in outgoing SH color		Low	None
(2) 3iGS [30]; [35]	Neural network	Net.-bounded	Jointly baked in the neural feature		Net.-bounded	None
(3) DeferredGS [36]	Physically-based rendering equation [14]	Arbitrary, thus bounded by the frequency of incident lighting	Ignored		None	Direct
(4) GShader [13]; [39]			Jointly baked in outgoing SH residual		Low	
(5) GS-ID [8]; [40]			Baked as SH	Ignored		
(6) GS-IR [18]; [10]				Baked as SH		
(7) GIR [29]			Ray-traced in 3D space	Baked as SH		
(8) GI-GS [2]			Ray-traced in screen space, incomplete		High	(In)direct
(9) Ours			Ray-traced in 3D space		High	(In)direct

first demonstrates efficient ray tracing in 3D Gaussians, enabling ray-based lighting effects such as non-projective views. RayGauss [1] and EVER [21] employ ray sampling in 3D Gaussians, mitigating common artifacts in 3DGS, such as aliasing [41], popping artifacts [27] and projection errors [12]. Yang *et al*. [46] adopt path tracing to simulate global illumination with Gaussian primitives. Meanwhile, Jorge *et al*. [4] utilizes ray sampling to simulate volumetric scattering and emissive media. Our SpecTRe-GS integrates ray tracing techniques to evaluate accurate incident radiance in the 3DGS framework at an affordable cost.

3. Proposed Method

Our method is specifically designed to accurately model the high-frequency reflections of highly specular surfaces. We build upon the recent advancements in 3DGS (Sec. 3.1) and adapt the reflection model to efficiently handle scenes comprising both highly specular objects and rougher objects (Sec. 3.2). To precisely and reliably query the incident radiance necessary for rendering both direct and indirect highly specular reflections, we incorporate ray tracing within the 3DGS framework (Sec. 3.3). We employ a dedicated training strategy to enhance geometry reconstruction, which is crucial for rendering undistorted and realistic reflections (Sec. 3.4). Our pipeline is outlined in Fig. 2.

3.1. Preliminary

The seminal work of 3DGS [15] represents the scene as a translucent Gaussian point cloud, where each 3D Gaussian is described with central point $\boldsymbol{\mu} \in \mathbb{R}^3$, scales $\boldsymbol{s} \in \mathbb{R}^3$, rotation quaternion $\boldsymbol{q} \in \mathbb{R}^4$, and opacity $\sigma \in [0, 1]$. The distribution of each Gaussian in 3D space is given by $G(\boldsymbol{x}) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$, where the covariance $\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^\top\mathbf{R}^\top$ is compactly parameterized as $\boldsymbol{s}$ and $\boldsymbol{q}$. By performing local affine transformation [48], 3D Gaussians are projected as 2D Gaussians on the image plane for rasterization. Then for each ray corresponding to an image pixel, the contributing 2D Gaussians are accumulated from front to back for volumetric rendering:

$$I(\boldsymbol{\omega}_\mathrm{o}) = \textstyle\sum_i \boldsymbol{c}_i(\boldsymbol{\omega}_\mathrm{o})\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \tag{1}$$

where $\alpha_i = \sigma_i G_i$ and the view-dependent color $\boldsymbol{c}_i(\boldsymbol{\omega}_\mathrm{o}) \in \mathbb{R}^3$ is modeled by third-order SH coefficients $\boldsymbol{\varphi}_i \in \mathbb{R}^{48}$.

Besides the view-dependent color $\boldsymbol{c}_i$, many other optimizable attributes including surface normal, diffuse albedo, or occlusion information, can be attached to each Gaussian. These attributes can be used locally in each Gaussian, or be accumulated to a map in the same way as Eq. (1) for later applications such as deferred shading [39].

3.2. Highly Specular Reflection Modeling

To model high-frequency reflections, we follow the convention to incorporate the physically-based rendering equation [14]. As commonly observed phenomena in real-world life, the inter-reflections from the nearby objects on highly specular surfaces should be modeled for a photorealistic scene appearance. The common practice with distant environment maps [13] ignores such inter-reflections and does not suffice for this task, thus we employ ray-tracing (detailed in Sec. 3.3) in 3D space for a complete and high-frequency inter-reflection. However, we found that performing ray tracing on the entire scene, including highly specular surfaces and rougher surfaces, is highly time-consuming. Evaluating the rendering equation [14] on rougher surfaces typically needs larger samples of the incident lighting even with multiple importance sampling [31], thus heavily increases the computational cost. Fortunately, previous works [15] have proven that rough surfaces can be well-captured by directly optimizing the outgoing color. Therefore, we propose the following reflection model that treats highly specular surfaces and rougher surfaces sepa-

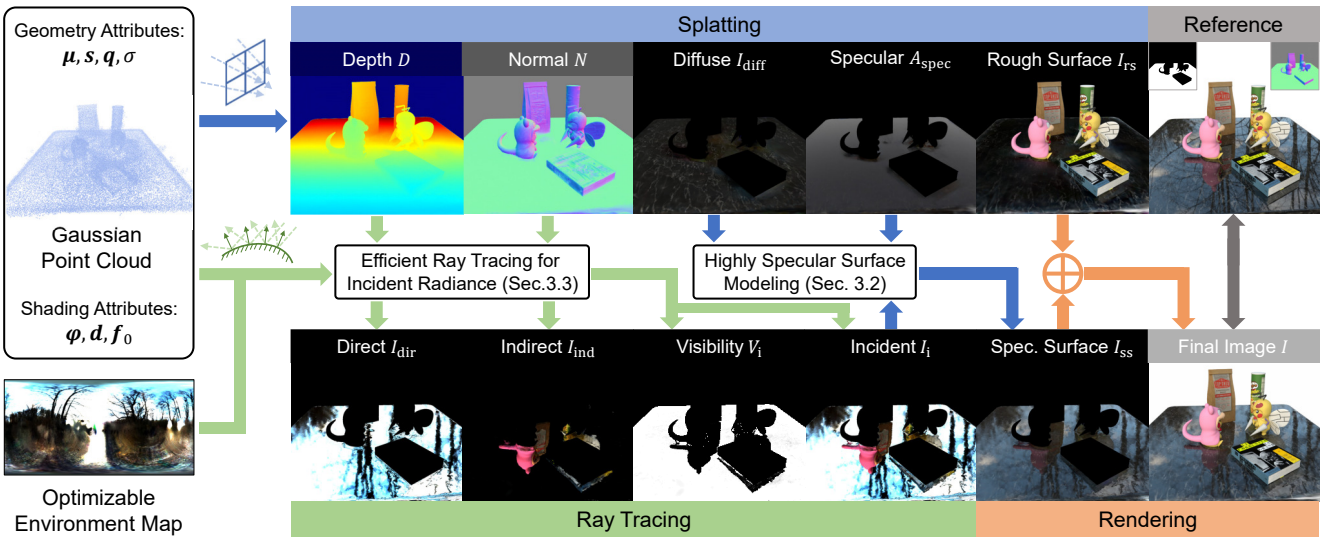

Figure 2. Overview of the SpecTRe-GS pipeline. The scene is represented as a Gaussian point cloud, where each Gaussian is associated with geometry and shading attributes used to render the depth map D, normal map N, diffuse reflection I_{diff}, specular reflectance A_{spec}, and rough surface reflection I_{rs} through splatting (Sec. 3.1). Highly specular surface reflection I_{ss} is formulated as the sum of the diffuse reflection I_{diff} and the specular reflection computed from the incident radiance I_{i} and specular reflectance A_{spec} (Sec. 3.2). The incident radiance I_{i} is determined along reflected secondary rays given by the scene geometry D and N, with the indirect component I_{ind} and visibility V_{i} evaluated via efficient ray tracing within the Gaussian point cloud, and the direct component I_{dir} modeled by an optimizable environment map (Sec. 3.3). The scene geometry is enhanced using normal prior guidance, numerical gradients from ray-traced incident radiance, and other training signals to achieve undistorted reflections (Sec. 3.4).

rately to enhance the specular reflections while maintaining the rendering speed. Also, we follow deferred shading [5] to shade for each pixel instead of each Gaussian, as it reduces the computation and gives sharp surface normals [47].

Modeling highly specular surfaces. To model surfaces that may exhibit both highly specular and diffuse reflections, such as polished marble, we define the reflection I_{ss} for these highly specular surfaces at surface point $\boldsymbol{x}$, observed from view direction $\boldsymbol{\omega}_{\text{o}}$ as follows:

$$I_{\text{ss}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{o}}) = I_{\text{diff}}(\boldsymbol{x}) + I_{\text{spec}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{o}}). \tag{2}$$

While the diffuse reflection I_{diff} is the view-independent outgoing radiance directly optimized as $\boldsymbol{d}$ in each Gaussian, the specular reflection I_{spec} is approximated with an ideal mirror reflection:

$$I_{\text{spec}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{o}}) = A_{\text{spec}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{r}}) I_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{r}}), \tag{3}$$

where the reflected incident direction $\boldsymbol{\omega}_{\text{r}}$ is given by $\boldsymbol{\omega}_{\text{o}} - 2\boldsymbol{\omega}_{\text{o}}^{\top} \boldsymbol{n} \cdot \boldsymbol{n}$ in which $\boldsymbol{n}$ denotes the local surface normal, and A_{spec} is the specular reflectance considering Fresnel effects through Schlick's approximation [28][1]:

$$A_{\text{spec}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{r}}) = F_0(\boldsymbol{x}) + (1 - F_0(\boldsymbol{x}))(1 - \boldsymbol{n}^{\top} \boldsymbol{\omega}_{\text{r}})^5. \tag{4}$$

Here, F_0 is the specular reflectance at normal incidence optimized as $\boldsymbol{f_0}$ in each Gaussian and is 3-channel to support metallic materials. The incident radiance I_{i} in $\boldsymbol{\omega}_{\text{r}}$ originates from both direct and indirect sources:

$$I_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{r}}) = V_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{r}}) I_{\text{dir}}(\boldsymbol{\omega}_{\text{r}}) + (1 - V_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{r}})) I_{\text{ind}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{r}}), \tag{5}$$

where $V_{\text{i}} \in [0, 1]$ quantifies the proportion of the incident lighting occluded by the scene itself, I_{dir} is the direct incident lighting from a distant environment map, and I_{ind} is the indirect incident radiance from the scene.

Defining depth and surface normal of Gaussians. It is essential to accurately acquire the depth and surface normal of each Gaussian in our method, as they directly determine the origin and direction of the reflection. However, they are ambiguous because the Gaussians are translucent ellipsoids in 3D space. Instead of treating the depth of the projected center and the shortest axis direction facing the camera as the depth and normal of each Gaussian [13, 17, 39], we adopt the formulation in GOF [42] to calculate for each Gaussian the "intersection point" along the ray achieving the maximum response, using its depth and normal of the intersection plane. This formulation gives a better and more consistent geometry reconstruction with 3D Gaussians.

Modeling the whole image. We model the part of the scene containing highly specular surfaces using Eq. (2). For the other part which exhibits only low-frequency view dependencies, we follow GOF [42] to model the appearance I_{rs} of the rougher surfaces using per-Gaussian SH coefficients $\boldsymbol{\varphi}$. We combine I_{ss} and I_{rs} using a binary mask M (which can be handily acquired through pre-trained segmentation models such as SAM [16]) specifying the highly specular

[1]We omit the distribution and mask-shadowing term in the microfacet model as we assume ideal mirror reflection.

surfaces within the scene ($\odot$ is pixel-wise multiplication):

$$I(\boldsymbol{\omega}_\mathrm{o}) = I_\mathrm{ss}(\boldsymbol{\omega}_\mathrm{o}) \odot M + I_\mathrm{rs}(\boldsymbol{\omega}_\mathrm{o}) \odot (1 - M). \quad (6)$$

3.3. Efficient Ray Tracing for Incident Radiance

To accurately compute the reflection from highly specular surfaces, it is crucial to obtain the incident radiance at the reflected view direction as modeled by Eq. (5), which incorporates both direct and indirect lighting. However, due to the incoherent origins and directions of these secondary rays, the incident radiance cannot be efficiently rendered using the rasterizer from the original 3DGS [15], as it relies on the locality of the rays, nor can it be computed merely using an environment map, which considers only direct lighting [13]. Inspired by the recent work 3DGRT [24], we implement an efficient ray tracer within the 3DGS framework to compute the radiance of secondary rays, thus achieving high-frequency and accurate reflections.

Geometry acceleration structure (GAS). For fast scene traversal, we construct a GAS within the OptiX programming interface [25], which uses ray tracing hardware to accelerate ray-primitive intersection tests and is several times faster than BVH [3] in CUDA. Instead of using axis-aligned bounding boxes which inefficiently encapsulate irregular Gaussians and lead to false positives, we adopt stretched icosahedrons as proxy meshes for more efficient ray intersection [24]. The vertex j of the icosahedron approximating Gaussian i is given by:

$$\boldsymbol{v}_{i,j} = \sqrt{2\log(\sigma_i/\alpha_\mathrm{min})}\mathbf{R}_i^\top \mathbf{S}_i^\top \boldsymbol{v}_j^\mathrm{unit} + \boldsymbol{\mu}_i, \quad (7)$$

where $\boldsymbol{v}_j^\mathrm{unit}$ is the vertex of an inscribed icosahedron in a unit sphere. Here, α_min serves as a threshold that adaptively scales each icosahedron by dictating the minimal contribution of the Gaussian at the transformed vertices, thereby reducing the computational overhead during intersection and radiance evaluation. We regularly update the GAS after each densification operation [15] and every n_up step as the Gaussian point cloud changes along the training.

Ray queries with GAS. To evaluate the incident radiance $I_\mathrm{ind}(\boldsymbol{x}_0, \boldsymbol{\omega}_\mathrm{r})$, secondary rays $\boldsymbol{x}(t) = \boldsymbol{x}_0 + t\boldsymbol{\omega}_\mathrm{r}$ are traced against proxy meshes within the GAS. Upon tracing each ray, we first collect the k closest valid hits, each corresponding to a Gaussian with $\alpha_i \geq \alpha_\mathrm{min}$. The depth of the "true" intersection point for each Gaussian is defined as:

$$t_{\mathrm{max},i} = \frac{(\boldsymbol{\mu}_i - \boldsymbol{x}_0)^\top \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\omega}_\mathrm{r}}{\boldsymbol{\omega}_\mathrm{r}^\top \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\omega}_\mathrm{r}}, \quad (8)$$

which maximizes the Gaussian's contribution $\sigma_i G_i(\boldsymbol{x}_0 + t_{\mathrm{max},i}\boldsymbol{\omega}_\mathrm{r})$. We then sort the Gaussians by their depth and update the ray depth D_ind, visibility (*i.e.*, transmittance) V_i, and color I_ind (using the Gaussians' outgoing SH colors assuming that they model rough surfaces), following Eq. (1). The above process is repeated after advancing the ray origin until no more valid hits remain or the ray transmittance drops below a threshold. For the transmittance that remains unoccluded by the reconstructed scene itself, we adopt an optimizable environment map for calculating the direct incident radiance $I_\mathrm{dir}(\boldsymbol{\omega})$. According to Eq. (5), the final incident radiance $I_\mathrm{i}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{r})$ combines direct and indirect components, weighted by visibility $V_\mathrm{i}(\boldsymbol{x}, \boldsymbol{\omega})$.

Thanks to the efficient and hardware-accelerated ray tracing techniques elaborated above, as well as our separate modeling of highly specular and rough surfaces, the average rendering time of our method is only 50% to 100% slower than that of a pure rasterizer on each of our test scenes. This varies according to the density of objects in the scene and the proportion of highly specular surfaces.

3.4. Training

We propose a training strategy that especially focuses on enhancing the geometry reconstruction, as unfaithful scene geometry induces inaccurate secondary rays and results in distorted reflections. We optimize our model from an initial sparse point cloud using the following loss terms:

$$\mathcal{L} = \mathcal{L}_\mathrm{c} + \lambda_\mathrm{d}\mathcal{L}_\mathrm{d} + \lambda_\mathrm{dn}\mathcal{L}_\mathrm{dn} + \lambda_\mathrm{sm}\mathcal{L}_\mathrm{sm} + \lambda_\mathrm{n}\mathcal{L}_\mathrm{n} + \lambda_\mathrm{r}\mathcal{L}_\mathrm{r}, \quad (9)$$

where $\mathcal{L}_\mathrm{c}$ is a color reconstruction loss [15] differentiable to the geometry, $\mathcal{L}_\mathrm{d}$ and $\mathcal{L}_\mathrm{dn}$ are the distortion and depth-normal consistency loss used in GOF [42], $\mathcal{L}_\mathrm{sm}$ is an edge-aware smoothness term for the normal and specular reflectance for the highly specular surfaces, $\mathcal{L}_\mathrm{n}$ is a normal prior term to employ data-driven clues, and $\mathcal{L}_\mathrm{r}$ is a residual-suppressing term for progressive learning. This section describes these terms and more details are in the supplement.

Normal prior guidance. To tackle the challenging problem of reconstructing geometry for highly specular objects, we employ StableNormal [38] to provide monocular normal priors $\hat{N}$, offering initial geometric constraints:

$$\mathcal{L}_\mathrm{n} = 1/|N| \sum ||(\hat{N} - N) \odot M||_1. \quad (10)$$

Joint geometry optimization. Recognizing potential inaccuracies and inconsistencies across views in the normal prior, we concurrently optimize the scene geometry during training. This is achieved by differentiating the incident radiance I_i, which relates to the final image through Eq. (2) and Eq. (3), with respect to the scene geometry. Computing the gradient of the direct incident radiance I_dir is straightforward. To compute the gradient of the indirect incident radiance I_ind concerning the local surface normal $\boldsymbol{n}$, we adopt a simple yet effective finite difference method to numerically evaluate the gradient of I_ind with respect to the reflected direction $\boldsymbol{\omega}$ that depends on $\boldsymbol{n}$:

$$\left.\frac{\partial I_\mathrm{ind}}{\partial \boldsymbol{\omega}^i}\right|_{\boldsymbol{\omega}_0} \approx \frac{I_\mathrm{ind}(\boldsymbol{x}, \boldsymbol{\omega}_0 + \Delta_{\boldsymbol{\omega}}^i) - I_\mathrm{ind}(\boldsymbol{x}, \boldsymbol{\omega}_0 - \Delta_{\boldsymbol{\omega}}^i)}{2\Delta_{\boldsymbol{\omega}}^i}, \quad (11)$$

where $\Delta_{\boldsymbol{\omega}}^i$ denotes the perturbation of the i-th dimension in $\boldsymbol{\omega}$. Considering that using the same perturbation for sec-

Table 2. Quantitative comparisons of novel view synthesis results with state-of-the-art methods on both synthetic and real-world scenes. We report the scores of PSNR, SSIM [34], and LPIPS [44] for entire images and within reflective regions. ↑ (↓) means higher (lower) is better. We mark the **best** and the second best results in each column.

Data	Synthetic (entire image)			Synthetic (reflective)			Real-world (entire image)			Real-world (reflective)		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS [15]	27.71	0.910	0.131	21.51	0.940	0.078	24.82	0.823	0.227	21.31	0.965	0.047
GOF [42]	27.97	0.917	0.117	21.38	0.940	0.076	24.86	0.826	0.219	21.25	0.963	0.047
GOF* [42]	27.57	0.913	0.125	20.67	0.936	0.085	24.69	0.825	0.223	20.44	0.962	0.051
3iGS [30]	28.58	0.912	0.125	21.98	0.939	0.076	24.37	0.809	0.233	21.22	0.963	0.048
GShader [13]	26.65	0.888	0.160	20.79	0.935	0.086	23.93	0.810	0.255	20.23	0.963	0.053
GShader* [13]	27.14	0.894	0.153	21.22	0.938	0.084	24.03	0.812	0.253	20.36	0.964	0.053
3DGS-DR [39]	27.62	0.890	0.173	21.57	0.938	0.085	24.79	0.824	0.243	21.18	0.965	0.050
3DGS-DR* [39]	26.97	0.883	0.182	21.18	0.936	0.088	24.60	0.823	0.251	20.38	0.964	0.055
Ours	**29.41**	**0.922**	**0.109**	**23.28**	**0.949**	**0.062**	**25.11**	**0.828**	**0.213**	**22.77**	**0.967**	**0.035**

ondary rays with different depths D_{ind} leads to different amounts of intersection movements, making the numerical gradient less accurate, we use a depth-aware ray perturbation that adjusts to D_{ind} for more informative gradients:

$$\Delta^i_{\boldsymbol{\omega}} = \theta_{\text{d}} \max(\min(D_{\text{ind}}/D_0, \sigma_{\text{d}}^{\max}), \sigma_{\text{d}}^{\min}), \tag{12}$$

where θ_{d} and D_0 define a reference angular perturbation and $\sigma_{\text{d}}^{\min}$ and $\sigma_{\text{d}}^{\max}$ ensure numerical stability. For computational efficiency, we disregard the gradient with respect to the point position $\boldsymbol{p}$, as we find reflections are distorted mainly due to normal errors. We detach visibility V_{ind} and depth D_{ind} because the sparsity of the visibility gradient and abrupt changes in depth damage numerical stability.

Progressive learning. As training progresses, monocular normal priors may hinder the accurate geometry reconstruction, thus we gradually decay the weight of normal prior guidance λ_{n}, letting gradients from physics-based rendering take over the fine-tuning of scene geometry. Besides, for smoother training and consequently better reconstruction within the reflective regions, we replace the hard mask M used in Eq. (6) with a relaxed version $1 - I_{\text{r}}$ rendered with another per-Gaussian attribute $r \in [0, 1]$ using Eq. (1). We encourage the reflective region to be modeled with highly specular reflections by suppressing the residual within M:

$$\mathcal{L}_{\text{r}} = 1/|M| \sum ||I_{\text{r}} \odot M||_1. \tag{13}$$

4. Experiments

In this section, we compare our method on both synthetic and real-world scenes (Sec. 4.2) against state-of-the-art methods (Sec. 4.3). We also perform ablation studies to validate the key components of our method (Sec. 4.4) and explore applications in scene editing (Sec. 4.5). Please check the supplementary material for more implementation details, data creation details, and experimental results.

4.1. Implementation Details

We implement our method in PyTorch [26] with CUDA kernels for rasterization based on GOF [42] and custom OptiX programs for ray tracing. We conduct all experiments on a GPU with 24 GB VRAM, 83 shader core TFLOPS, and 191 ray tracing core TFLOPS. We set $\lambda_{\text{sm}} = 0.5$, $\lambda_{\text{r}} = 1.0$, and $\lambda_{\text{n}} = 0.5$, exponentially decaying to 0.0025 from 4k to 10k iterations. We use $r = 0.9$ as the initialization. For other hyperparameters, we use the settings in GOF [42].

4.2. Datasets

Synthetic scenes. We build 6 synthetic scenes, each containing multiple surrounding objects and a highly specular surface with distinctive properties: HELMET, MARBLETABLE, VASE, POT, TOASTER, and MIRROR. For each scene, we use the Blender Cycles engine [6] to render about 300 images of resolution 1024 × 1024, of which 2/3 are used for training and the rest are left for testing.

Real-world scenes. For real-world validation, we capture two scenes: REALBOWL and REALPOT, consisting of 138 and 139 images of resolution 1440 × 1080 respectively, with every eighth image reserved in the test set.

4.3. Comparison

Comparing methods. We compare our method with 5 open-source state-of-the-art methods: the original 3DGS [15]; GOF [42] which focus on geometry reconstruction with 3D Gaussians; 3iGS [30] which improves view-dependent effects through neural features; and 3DGS-DR [39] and GShader [13] which are adapted for specular reflection employing physically-based rendering (PBR). For a fair comparison, we provide GOF [42], 3DGS-DR [39], and GShader [13] with the normal prior guidance, and indicate these versions by " * " throughout this paper.

Evaluation protocol. We report peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM) [34], and learned perceptual image patch similarity (LPIPS) [44] measured on both entire images and reflective regions.

Quantitative evaluation. As shown in Tab. 2, our method outperforms comparing methods on both synthetic and real-world scenes, especially in reflective regions. It is worth noting that using normal prior guidance usually leads to degradation of GOF [42] which relies solely on SH color, as it suppresses the ability to simulate high-frequency appear-

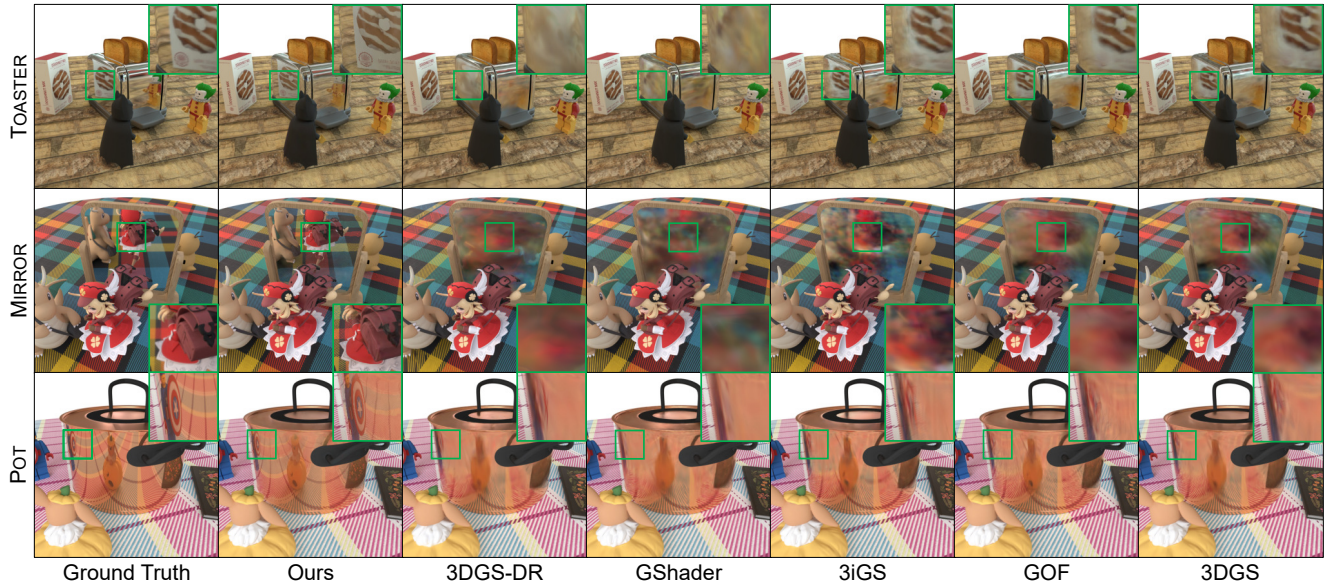

Figure 3. Novel view synthesis comparison on synthetic scenes. We evaluate SpecTRe-GS against relevant 3DGS-based methods modeling view-dependent appearances: 3DGS-DR [39], GShader [13], 3iGS [30], GOF [42], and 3DGS [15]. The insets highlight the improvement of SpecTRe-GS in rendering high-frequency specular reflections, showcasing the precise positioning of reflected content and sharp details.

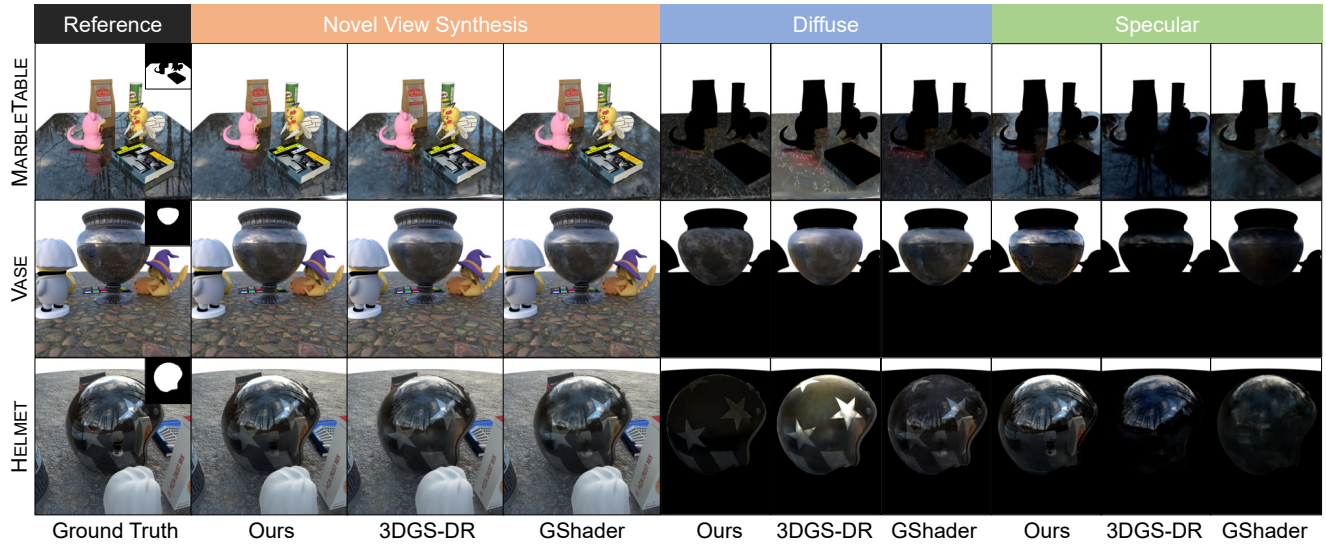

Figure 4. Novel view synthesis and image decomposition results on synthetic scenes, comparing our SpecTRe-GS with 3DGS-DR [39] and GShader [13]. SpecTRe-GS better distinguishes diffuse and specular components in reflective regions, with the least baked-in view-dependent appearance in diffuse components. The masks of reflective regions are shown as insets in the reference image.

ances through false geometry. In contrast, the influence of this guidance on 3DGS-DR [39] and GShader [13] is two-fold: It facilitates the reconstruction of appearance under direct lighting, but also prevents these methods from imitating specular inter-reflections by faking geometry.

Qualitative evaluation. In visual comparisons, we show results of GOF [42], GShader* [13], and 3DGS-DR [39] as they are variants with higher qualities. As shown in Fig. 3, our method performs better than the baseline methods for both planar reflectors (rows 1-2) and curved reflectors (row 3), capturing clean, accurate, and high-frequency specular reflections. Our method also surpasses baseline methods in real-world scenes, as shown in Fig. 1. By visualizing the separate components of PBR-based methods in Fig. 4, we find that incident radiance modeling for both direct and indirect sources is vital for physically meaningful scene appearance decomposition. Otherwise, to fit the overall color in training images, indirect incident radiance is heavily baked into the diffuse component (row 1 for 3DGS-DR [39] and row 3 for GShader [13]). The baked-in incident radiance loses high-frequency details and prevents realistic scene editing in a physically-accurate manner. It should be noted

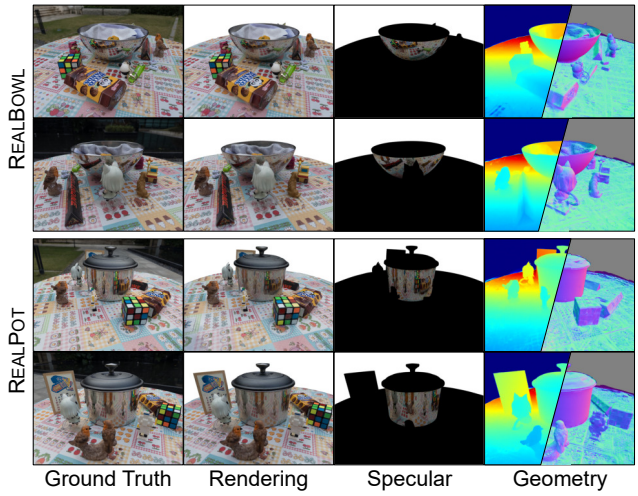

Figure 5. Our reconstruction results on two real-world scenes. SpecTRe-GS accurately reconstructs the geometry of highly specular surfaces alongside the reflection of nearby objects.

that 3DGS-DR [39] performs better than GShader [13] in reconstructing the direct part of incident radiance from the environment, showing the superiority of surface-based deferred shading over volumetric rendering in PBR modeling of high-frequency reflections. Our method, with well-optimized geometry and consequent accurate incident radiance modeling, achieves satisfactory novel view synthesis of specular reflection on both synthetic and real-world scenes, as also indicated in Fig. 5.

4.4. Ablation Study

On a subset of our synthetic scenes, we validate the following key component of our method: (1) **N**ormal prior guidance; (2) **J**oint geometry optimization from incident lighting; (3) ray-traced **I**ndirect incident radiance; (4) **P**rogressive learning; and (5) **D**epth-aware ray perturbation. The results are shown in Tab. 3. Our full method (Ours) achieves the highest scores, confirming the effectiveness of our method designs. Without the initial normal prior guidance (Ours w/o **N**.), the optimization process is prone to be trapped into local minima far away from the true underneath geometry of the reflective region. On the other hand, only relying on normal priors without further joint optimization for fine-tuning geometry (Ours w/o **J**.) also fails in reconstructing accurate geometry for physically-based high-frequency reflections. As discussed above, using only direct incident lighting modeling (Ours w/o **I**.) leads to baked-in indirect incident lighting and damages the quality of high-frequency components. The progressive learning scheme (Ours w/o **P**.) and depth-aware ray perturbation (Ours w/o **D**.) also further helps to faithfully reconstruct the scene.

4.5. Applications

Our Gaussian point cloud representation with explicit geometry, material, and incident lighting facilitates various editing applications. As illustrated in Fig. 6, specular objects of interest can be faithfully captured as assets in the presence of unwanted reflections from nearby objects, enabling further usages such as relighting (column 1) and material editing (column 4). We can also manipulate the scene geometry such as deletion (column 2) and insertion (column 3), with specular inter-reflections adjusted accordingly.

Table 3. Ablation study with variants of our proposed method excluding: **N**ormal prior guidance, **J**oint geometry optimization, **I**ndirect incident lighting modeling, **P**rogressive learning, and **D**epth-aware ray perturbation. ↑ (↓) means higher (lower) is better.

Method	Entire Image			Reflective Region		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Ours w/o **N**.	26.70	0.885	0.149	19.80	0.913	0.094
Ours w/o **J**.	26.37	0.878	0.154	19.52	0.907	0.100
Ours w/o **I**.	28.49	0.905	0.130	22.19	0.935	0.074
Ours w/o **P**.	29.26	0.910	0.118	23.16	0.940	0.062
Ours w/o **D**.	29.75	**0.914**	0.114	23.88	**0.944**	**0.056**
Ours	**29.90**	**0.914**	**0.112**	**24.05**	**0.944**	**0.056**

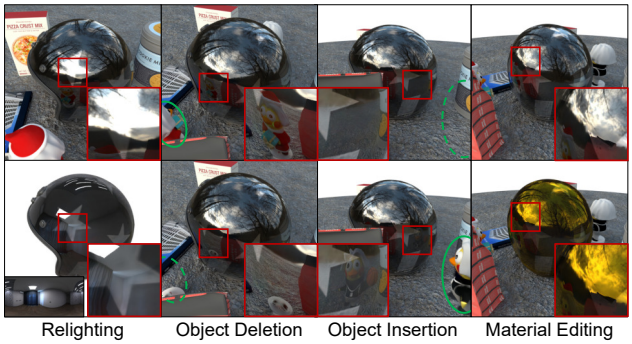

Figure 6. Our explicit representation of geometry, material, and incident lighting in the Gaussian point cloud enables various editing applications. Insets have been brightened for clarity.

5. Conclusion

We introduce SpecTRe-GS to model highly specular surfaces with inter-reflections, distinguished by its separate modeling of specular and rough surfaces, the efficient ray tracer, and a geometry-enhancing training strategy for fast and accurate specular inter-reflection rendering.

Limitations. We only model specular inter-reflections from rough surfaces to avoid multi-bounce path tracing, and we only trace one secondary ray for each pixel in the reflective region, approximating the specular reflection with an ideal mirror. Also, the monocular normal prior may fail in the existence of strong semantic ambiguity (such as mirrors with hidden borders), making our geometry fail to initialize.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 62136001. PKU-affiliated authors thank *openbayes.com* for providing computing resource.

References

[1] Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. RayGauss: Volumetric Gaussian-based ray casting for photorealistic novel view synthesis. In *Proc. of IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2024. 3

[2] Hongze Chen, Zehong Lin, and Jun Zhang. GI-GS: Global illumination decomposition on Gaussian splatting for inverse rendering. In *Proc. of International Conference on Learning Representations (ICLR)*, 2025. 1, 2, 3

[3] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM (CACM)*, 19(10):547–554, 1976. 5

[4] Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. Don't splat your Gaussians: Volumetric ray-traced primitives for modeling and rendering scattering and emissive media. *ACM Transactions on Graphics (TOG)*, 44(1):10:1–10:17, 2025. 3

[5] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a VLSI system for high performance graphics. In *Proc. of the ACM SIGGRAPH Conference and Exhibition On Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1988. 4

[6] Cycles Developers. Cycles: Open source production rendering, 2024. https://www.cycles-renderer.org/. Accessed: 2024-11-06. 6

[7] Jan-Niklas Dihlmann, Arjun Majumdar, Andreas Engelhardt, Raphael Braun, and Hendrik P. A. Lensch. Subsurface scattering for 3D Gaussian splatting. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 2024. 1

[8] Kang Du, Zhihao Liang, and Zeyu Wang. GS-ID: Illumination decomposition on Gaussian splatting via diffusion prior and parametric light source optimization. In *arXiv preprint arXiv:2408.08524*, 2024. 1, 2, 3

[9] Fan Fei, Jiajun Tang, Ping Tan, and Boxin Shi. VMINer: Versatile multi-view inverse rendering with near- and far-field light sources. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[10] 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 *Proc. of European Conference on Computer Vision (ECCV)*, 2024. 1, 2, 3

[11] Yijia Guo, Yuanxi Bai, Liwen Hu, Ziyi Guo, Mianzhi Liu, Yu Cai, Tiejun Huang, and Lei Ma. PRTGS: Precomputed radiance transfer of Gaussian splats for real-time high-quality relighting. In *Proc. of ACM International Conference on Multimedia (ACM MM)*, 2024. 1

[12] Letian Huang, Jiayang Bai, Jie Guo, Yuanqi Li, and Yanwen Guo. On the error analysis of 3D Gaussian splatting and an optimal projection strategy. In *Proc. of European Conference on Computer Vision (ECCV)*, 2024. 3

[13] 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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 3, 4, 5, 6, 7, 8

[14] James T. Kajiya. The rendering equation. In *Proc. of the ACM SIGGRAPH Conference and Exhibition On Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1986. 2, 3

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):139:1–139:14, 2023. 1, 2, 3, 5, 6, 7

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 4

[17] 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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 4

[18] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. GS-IR: 3D Gaussian splatting for inverse rendering. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 3

[19] Jiayue Liu, Xiao Tang, Freeman Cheng, Roy Yang, Zhihao Li, Jianzhuang Liu, Yi Huang, Jiaqi Lin, Shiyong Liu, Xiaofei Wu, Songcen Xu, and Chun Yuan. MirrorGaussian: Reflecting 3D Gaussians for reconstructing mirror reflections. In *Proc. of European Conference on Computer Vision (ECCV)*, 2024. 2

[20] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. NeRO: Neural geometry and BRDF reconstruction of reflective objects from multiview images. *ACM Transactions on Graphics (TOG)*, 42(4):114:1–114:22, 2023. 2

[21] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T. Barron, and Yinda Zhang. EVER: Exact volumetric ellipsoid rendering for real-time view synthesis. In *arXiv preprint arXiv:2410.01804*, 2024. 3

[22] Jiarui Meng, Haijie Li, Yanmin Wu, Qiankun Gao, Shuzhou Yang, Jian Zhang, and Siwei Ma. Mirror-3DGS: Incorporating mirror reflections into 3D Gaussian splatting. In *Proc. of IEEE International Conference on Visual Communications and Image Processing*, 2024. 2

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of European Conference on Computer Vision (ECCV)*, 2020. 1

[24] Nicolas Moënne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3D Gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics (TOG)*, 43(6):232:1–232:19, 2024. 2, 5

[25] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David P. Luebke, David K. McAllister, Morgan McGuire, R. Keith Morley, Austin Robison, and Martin Stich. OptiX: a general purpose ray tracing engine. In *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 2010. 5

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. 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 *Proc. of Neural Information Processing Systems (NeurIPS)*, 2019. 6

[27] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian splatting for view-consistent real-time rendering. *ACM Transactions on Graphics (TOG)*, 43(4):64:1–64:17, 2024. 3

[28] Christophe Schlick. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum (CGF)*, 13(3):233–246, 1994. 4

[29] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jingtuo Liu, Liangjun Zhang, Jian Zhang, Bin Zhou, Errui Ding, and Jingdong Wang. GIR: 3D Gaussian inverse rendering for relightable scene factorization. In *arXiv preprint arXiv:2312.05133*, 2023. 1, 2, 3

[30] Zhe Jun Tang and Tat-Jen Cham. 3iGS: Factorised tensorial illumination for 3D Gaussian splatting. In *Proc. of European Conference on Computer Vision (ECCV)*, 2024. 1, 2, 3, 6, 7

[31] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proc. of the ACM SIGGRAPH Conference and Exhibition On Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1995. 3

[32] 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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[33] Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ben Mildenhall, Benjamin Attal, Richard Szeliski, and Jonathan T. Barron. NeRF-Casting: Improved view-dependent appearance with consistent reflections. In *Proc. of the ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia (SIGGRAPH Asia)*, 2024. 2

[34] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004. 6

[35] Zhiru Wang, Shiyun Xie, Chengwei Pan, and Guoping Wang. SpecGaussian with latent features: A high-quality modeling of the view-dependent appearance for 3D Gaussian splatting. In *Proc. of ACM International Conference on Multimedia (ACM MM)*, 2024. 1, 2, 3

[36] Tong Wu, Jia-Mu Sun, Yu-Kun Lai, Yuewen Ma, Leif Kobbelt, and Lin Gao. DeferredGS: Decoupled and editable Gaussian splatting with deferred shading. In *arXiv preprint arXiv:2404.09412*, 2024. 1, 2, 3

[37] Wangze Xu, Huachen Gao, Shihe Shen, Rui Peng, Jianbo Jiao, and Ronggang Wang. MVPGS: Excavating multi-view priors for Gaussian splatting from sparse input views. In *Proc. of European Conference on Computer Vision (ECCV)*, 2024. 1

[38] Chongjie Ye, Lingteng Qiu, Xiaodong Gu, Qi Zuo, Yushuang Wu, Zilong Dong, Liefeng Bo, Yuliang Xiu, and Xiaoguang Han. StableNormal: Reducing diffusion variance for stable and sharp normal. In *Proc. of the ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia (SIGGRAPH Asia)*, 2024. 5

[39] Keyang Ye, Qiming Hou, and Kun Zhou. 3D Gaussian splatting with deferred reflection. In *Proc. of the ACM SIGGRAPH Conference and Exhibition On Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2024. 1, 2, 3, 4, 6, 7, 8

[40] Keyang Ye, Qiming Hou, and Kun Zhou. Progressive radiance distillation for inverse rendering with Gaussian splatting. In *arXiv preprint arXiv:2408.07595*, 2024. 1, 2, 3

[41] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian splatting. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[42] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):271:1–271:13, 2024. 1, 2, 4, 5, 6, 7

[43] Libo Zhang, Yuxuan Han, Wenbin Lin, Jingwang Ling, and Feng Xu. PRTGaussian: Efficient relighting using 3D Gaussians with precomputed radiance transfer. In *Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2024. 1

[44] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[45] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[46] Yang Zhou, Songyin Wu, and Ling-Qi Yan. Unified Gaussian primitives for scene representation and rendering. In *arXiv preprint arXiv:2406.09733*, 2024. 3

[47] Zuo-Liang Zhu, Beibei Wang, and Jian Yang. GS-ROR: 3D Gaussian splatting for reflective object relighting via SDF priors. In *arXiv preprint arXiv:2406.18544*, 2024. 1, 2, 4

[48] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. EWA volume splatting. In *Proc. of IEEE Visualization Conference (VIS)*, 2001. 3

SpecTRe-GS: Modeling Highly Specular Surfaces with Reflected Nearby Objects by Tracing Rays in 3D Gaussian Splatting

Supplementary Material

Jiajun Tang[1,2] Fan Fei[1,2] Zhihao Li[3] Xiao Tang[3] Shiyong Liu[3]
Youyu Chen[4] Binxiao Huang[5] Zhenyu Chen[3] Xiaofei Wu[3] Boxin Shi[1,2#]

[1]State Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University
[2]National Engineering Research Center of Visual Technology, School of Computer Science, Peking University
[3]Huawei Noah's Ark Lab [4]Harbin Institute of Technology [5]University of Hong Kong

In this supplementary material, we provide additional implementation details for SpecTRe-GS (Sec. 6), the creation details of synthetic and real-world data (Sec. 7), and further qualitative/quantitative results and experimental analyses (Sec. 8). The videos on the project page[1] showcase qualitative view interpolation and scene editing results.

6. Additional Implementation Details

6.1. Implementation of Ray Tracing

The ray tracer described in Sec. 3.3 in the main paper is implemented in OptiX 7 [7] and integrated into the 3DGS framework with the PyOptiX package as the Python bindings for the OptiX host API. According to the OptiX 7 specification, the OptiX pipeline consists of user-programmable entry points (programs) for different stages in ray tracing, we implement ray tracing in the Gaussians point cloud with custom *ray-gen* and *any-hit* programs:

- we initialize ray origin and direction, set and read the per-ray payloads, evaluate responses of Gaussians, and aggregate the volumetric radiance in the *ray-gen* program;
- we calculate the precise hit point of the mesh-bounded Gaussians and store the hit information into the per-ray buffer in the *any-hit* program.

The programs are described in Proc. 1 and Proc. 2. In our implementation, we construct a max heap of size $k = 256$ to store the closest hits with the complexity of $\mathcal{O}(N \log k)$, where N is the total number of hits.

6.2. Alignment of Rasteriazion and Ray Tracing

We align the radiance aggregation process of our ray tracer to that of the rasterizer. We use the rasterizer in GOF [14] as GOF calculates the maximum response of Gaussians along the rays. However, GOF still uses projected center depths to sort Gaussians in volumetric rendering. Therefore, we deliberately calculate projected depths as $t_{\text{hit},i}$ in our ray tracer and use $t_{\text{hit},i}$ instead of $t_{\text{max},i}$ for the ordering in Eq. (1). We also use the same ray termination threshold of remaining transmittance $T_{\min} = 0.001$ and max number of contributing Gaussians $K_{\max} = 256$ for our rasterizer and ray tracer. This reduces the discrepancy between the rendered appearances of the same Gaussian point cloud by these two renderers, which would cause the inconsistency between the directly observed appearance of objects and their appearance through a highly specular surface.

Procedure 1: *Ray-gen* Program

Input: ray origin $\boldsymbol{o}$, ray direction $\boldsymbol{d}$, GAS handle $\mathcal{H}$, Gaussians $\{\mathcal{G}_i\}$, min transmittance $T_{\min}$, min contribution $\alpha_{\min}$, hit buffer size k, min ray distance t_{near}, max ray distance t_{far}

Output: ray incident radiance I_{ind}, ray visibility V_{i}, ray depth D_{ind}

$I_{\text{ind}} \leftarrow (0,0,0)$;
$V_{\text{i}} \leftarrow 1$;
$D_{\text{ind}} \leftarrow 0$;
$t_{\text{curr}} \leftarrow t_{\text{near}}$;
while $t_{\text{curr}} < t_{\text{far}}$ **and** $V_{\text{i}} > T_{\min}$ **do**
  $c \leftarrow 0$;
  $\mathcal{B} \leftarrow \text{buffer}(k)$;
  $\text{setPayload}(\mathcal{B}, c)$;
  $\text{traceRay}(\mathcal{H}, \boldsymbol{o} + t_{\text{curr}}\boldsymbol{d}, \boldsymbol{d}, k)$;
  $\mathcal{B}, c \leftarrow \text{getPayload}()$;
  if $c = 0$ **then**
    $\text{terminateRay}()$;
  end
  $\mathcal{B} \leftarrow \text{sort}(\mathcal{B})$;
  for (t_{hit}, i) **in** $\mathcal{B}$ **do**
    $t_{\max}, \alpha_{\text{hit}} \leftarrow \text{response}(\boldsymbol{\mu}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{o}, \boldsymbol{d})$;
    if $\alpha_{\text{hit}} > \alpha_{\min}$ **then**
      $\boldsymbol{c}_i = \text{SH}(\boldsymbol{\varphi}_i, \boldsymbol{d})$;
      $I_{\text{ind}} \leftarrow I_{\text{ind}} + \alpha_{\text{hit}} V_{\text{i}} \boldsymbol{c}_i$;
      $D_{\text{ind}} \leftarrow D_{\text{ind}} + \alpha_{\text{hit}} V_{\text{i}} t_{\max}$;
      $V_{\text{i}} \leftarrow (1 - \alpha_{\text{hit}}) V_{\text{i}}$;
    end
    $t_{\text{curr}} \leftarrow t_{\text{hit}}$;
  end
end

The work was done during an internship in Huawei.
#Corresponding author. E-mail: shiboxin@pku.edu.cn.
[1]https://spectre-gs.github.io/

Procedure 2: *Any-hit* Program

Input: ray origin $\boldsymbol{o}$, ray direction $\boldsymbol{d}$, Gaussians $\{\mathcal{G}_i\}$, hitted primitive index i, hit buffer $\mathcal{B}$, hit buffer size k, hit count c

Output: in-place modified hit buffer $\mathcal{B}$, hit count c

$t_{\text{hit}} \leftarrow \text{projectDepth}(\boldsymbol{\mu}_i, \boldsymbol{o}, \boldsymbol{d})$;
$h \leftarrow (t_{\text{hit}}, i)$;
if $c = k$ **then**
  $h_{\text{max}} \leftarrow \mathcal{B}.\text{popMax}()$;
else
  $h_{\text{max}} \leftarrow (+\infty, -1))$;
  $c \leftarrow c + 1$;
end
$h_{\text{new}} \leftarrow \text{findCloser}(h_{\text{max}}, h)$;
$\mathcal{B}.\text{insert}(h_{\text{new}})$;

6.3. Training Details

During training, we use the same color reconstruction loss as commonly adopted in 3DGS-based methods [5]:

$$\mathcal{L}_{\text{c}} = 0.8 \cdot \frac{1}{|I_{\text{GT}}|} \sum ||I_{\text{GT}} - I||_1 - 0.2 \cdot \text{SSIM}(I_{\text{GT}}, I), \tag{14}$$

and we also apply this loss to I_{ss} in later training steps (>15k iterations) to encourage physics-based modeling of highly reflective regions.

We follow Eq. (1) to compute the mean depth of contributing Gaussians as the surface depth, instead of the depth of the "median" Gaussian in GOF [14], which we find is generally more noisy and inefficient in utilizing gradient signals. In the first 4k steps, we only rely on SH color modeling to quickly get a rough geometry initialization and low-frequency view-dependent radiance modeling.

Since the Fresnel reflectance is calculated from approximation (Eq. (4)), we detach $\partial A_{\text{spec}} / \partial \boldsymbol{n}$ and clip A_{spec} as $\min(A_{\text{spec}}, 10F_0)$ to ensure numerical stability.

We only run StableNormal [12] once for all scenes and save the estimated normals as monocular normal priors.

6.4. Tone Mapping

Our method operates in linear color space as required by physically-based rendering. We assume the gamma-corrected sRGB space of $\gamma = 2.2$ is usually used in the input images, which is closer to human perception. Thus, we can convert the images into linear color space by inversely applying the gamma correction. We convert our results back to the commonly adopted gamma-corrected sRGB space with $\gamma = 2.2$ prior to visualization or the computation of photometric losses and error metrics.

7. Data Creation Details

7.1. Synthetic Scenes

We collect 6 synthetic scenes using the Blender Cycles engine [3]: HELMET, MARBLETABLE, VASE, POT, TOASTER, and MIRROR, as described in Sec. 4.2 in the main paper. We show example images of each scene in Tab. 4 and Tab. 5 of this document.

7.2. Real-world Scenes

For real-world scenes, we use a hand-held iPhone 15 Pro and the "ProCam" app to take raw images with a linear camera response. We fix the white balance, focal length, exposure time, and ISO for all images in the same scene. We register the camera poses using COLMAP [8, 9] with SuperPoint [2] for feature extraction and LightGlue [6] for matching. After obtaining the captured raw images of the scene, we use a custom image signal processor (ISP) to process the raw image by, *e.g.*, demosaicking, white balancing, transforming color space, and most importantly, applying a tone mapping with $\gamma = 2.2$ to let the processed images satisfy our assumption of availability of linear space images. We resize the images to 1440×1080 and remove the out-of-focus background regions. The highly reflective regions are manually marked. By doing so, we collect 2 real-world scenes: REALBOWL and REALPOT. We show example images of each scene in Tab. 5 of this document.

8. Additional Results

8.1. Results with Varying Roughness

Our method is designed for perfect mirror reflections. Nevertheless, the inclusion of a low-frequency component gives it the ability to model rougher surfaces to some extent. Fig. 7 shows its results on the HELMET scene with varying roughness, from highly smooth ($\rho = 0.05$) to medium rough ($\rho = 0.3$). For each roughness value, we show the rendered image and the ground truth image in a test view, alongside the decomposition of specular component $A_{\text{spec}} I_{\text{i}} \odot (1 - I_{\text{r}})$ and low-frequency component $I_{\text{diff}} \odot (1 - I_{\text{r}}) + I_{\text{rs}} \odot I_{\text{r}}$, expanded according to the modeling in Sec. 3.2 and soft mask I_{r} in Sec. 3.4. As the roughness increases, while direct specular reflections can be approximated by blurred environment maps, indirect specular reflections in the lower half of the helmet are mimicked by brighter I_{rs} with lower A_{spec} values.

8.2. Geometry Representation

As shown in Fig. 8, our method can better capture the planar surface in MARBLETABLE scene with most points well aligned to the object, benefitting from our normal prior guidance and joint optimization of incident radiance and geometry. Without accurate geometry optimization and inci-

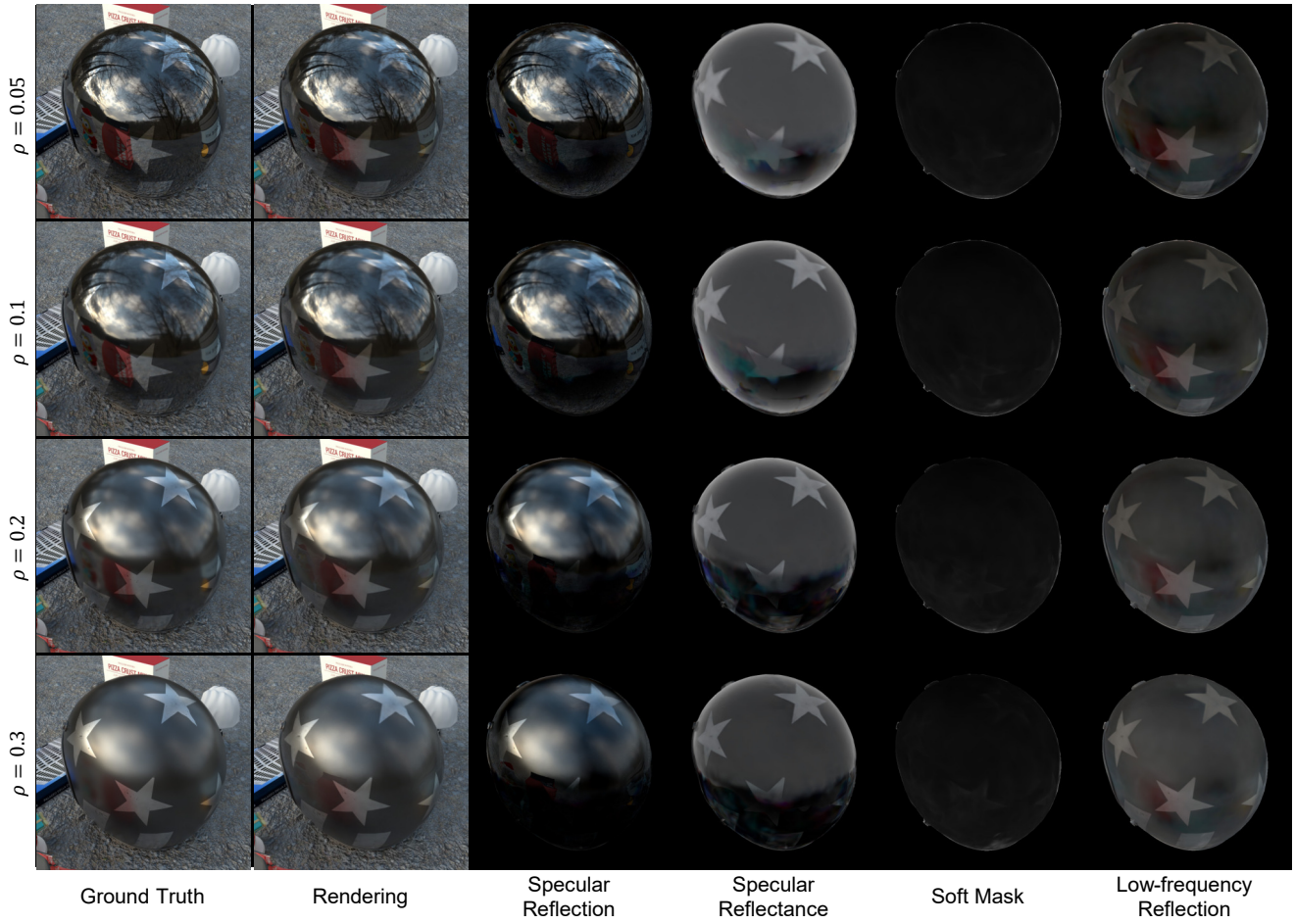

Figure 7. As surface roughness increases, our method attributes more proportion of the scene appearance to low-frequency reflection instead of perfect mirror reflection.

dent radiance reconstruction, other methods tend to fit high-frequency view-dependent specular reflection with high-frequency floaters.

8.3. Qualitative Ablation Results

Fig. 9 shows the results of the ablated variants of our method mentioned in Sec. 4.4 in the same view as Fig. 4. When the monocular normal prior guidance is absent during early training stages (Ours w/o **N**.), the training loss terms tend to overemphasize color reconstruction fidelity in observed images, causing the scene representation to converge to local minima with geometry deviating from ground truth in highly specular regions (as shown by the translucent artifacts on the left side of the helmet in column 1, indicating incomplete underlying geometric reconstruction). Conversely, when relying solely on monocular normal priors without subsequent joint optimization to refine scene geometry (Ours w/o **J**.), the inherent inaccuracies and multi-view inconsistencies in monocular normal predictions prevent the reconstruction of precise geometry required for physics-based high-frequency reflection modeling (evidenced by the missing high-frequency details in the reflections on the helmet surface, as depicted in column 2). As previously discussed regarding physics-based rendering approaches, modeling only direct illumination (Ours w/o **I**.) leads to indirect lighting effects being baked into either SH color representations or diffuse albedo, thereby compromising high-frequency component quality (manifested as missing high-frequency details and ghosting artifacts in the lower helmet region due to inter-reflections, shown in column 3). The progressive learning scheme (Ours w/o **P**.) and depth-aware ray perturbation (Ours w/o **D**.) also significantly contribute to faithful reconstruction of view-dependent high-frequency specular reflections.

8.4. Alignment of Rendering Methods

We analyze the consistency of rendering results from our rasterizer and ray tracer on the STUMP scene of the Mip-NeRF 360 dataset [1], which is a prerequisite for accurately evaluating indirect incident radiance. We show the rendered images and the corresponding PSNR scores of each renderer in Fig. 10, accompanied by the error map visualiza-

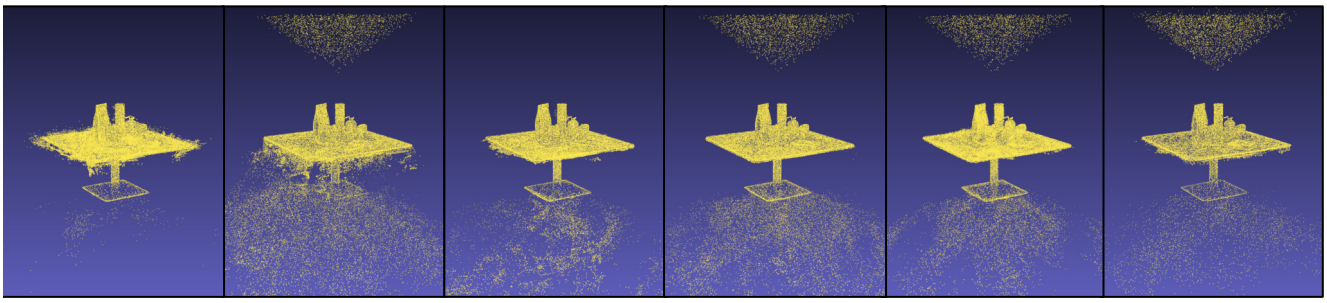

Figure 8. Visualization of the point cloud reconstructed by comparing methods [4, 5, 10, 13, 14]. Ours more faithfully captures the underlying geometry of reflective regions, while other methods disrupt their geometry to imitate highly specular reflections.

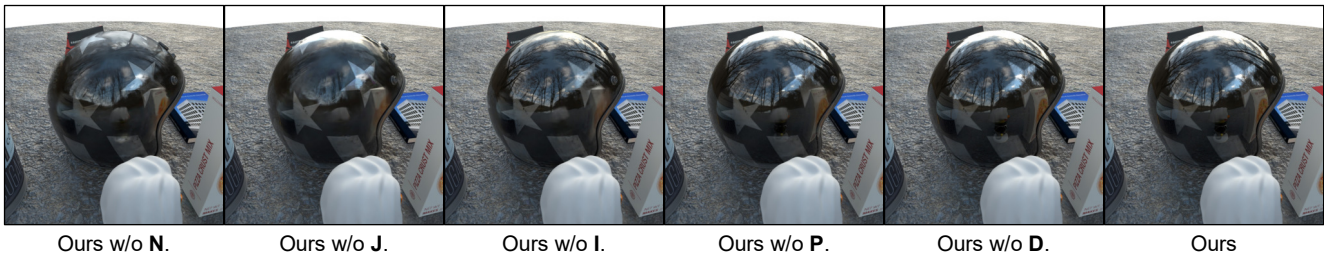

Figure 9. Qualitative ablation results with variants of our proposed method excluding: **N**ormal prior guidance, **J**oint geometry optimization, **I**ndirect incident lighting modeling, **P**rogressive learning, and **D**epth-aware ray perturbation.

Figure 10. Rasterized and ray-traced results in our proposed method are highly consistent, which ensures accurate indirect incident radiance queries from the Gaussian point cloud shared by our rasterizer and ray tracer.

tion. The rendering results from those two renderers in our pipeline remain highly consistent, as indicated by the inconspicuous visual difference, the close PSNR scores, and the colors in the error map.

8.5. More Quantitative Results

We show detailed quantitative results on each scene in Tab. 4 and Tab. 5 of this document. In general, our SpecTRe-GS consistently outperforms most compared methods on both synthetic scenes and real-world scenes, especially within reflective regions.

8.6. More Qualitative Results

We show additional qualitative comparisons with the baseline methods[2] on each scene in Fig. 11-14 of this document. For each scene, we show comprehensive visual comparison results from multiple test views.

[2]We show results of GShader* for all scenes, 3DGS-DR* for HELMET scene as their better performance indicated by quantitative evaluations.

We provide videos of view interpolation results as attached files on the project page. Compared with baseline methods, our method gives more view-consistent renderings of high-frequency reflection, which better respects the geometry of the highly reflective surfaces. In addition, we provide videos of the scene editing results.

Table 4. Quantitative comparison results with state-of-the-art methods on each of the 4 synthetic scenes (HELMET, MARBLETABLE, VASE, and POT). We show example images and the dataset splits of each scene in the leftmost column. We report the scores of PSNR, SSIM [11], and LPIPS [15] for entire images and within reflective regions. We mark the **best** and the second best results in each column. ↑ (↓) means higher (lower) is better.

Scene	Method	Entire Image			Reflective Region		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HELMET (train: 201, test:100) 	3DGS [5]	27.92	0.881	0.157	21.62	0.918	0.090
	GOF [14]	28.28	0.893	0.130	21.62	0.918	0.086
	GOF* [14]	27.15	0.887	0.145	20.28	0.912	0.100
	3iGS [10]	28.33	0.883	0.152	22.06	0.919	0.088
	GShader [4]	25.80	0.836	0.204	20.72	0.913	0.098
	GShader* [4]	26.51	0.846	0.196	21.28	0.915	0.097
	3DGS-DR [13]	26.32	0.841	0.233	20.76	0.914	0.098
	3DGS-DR* [13]	27.15	0.845	0.226	22.36	0.927	0.083
	Ours	**29.90**	**0.914**	**0.112**	**24.05**	**0.944**	**0.056**
MARBLETABLE (train: 233, test:123) 	3DGS [5]	22.41	0.858	0.197	20.10	0.889	0.142
	GOF [14]	23.59	0.873	**0.177**	19.77	0.889	**0.137**
	GOF* [14]	24.14	0.870	0.187	19.71	0.884	0.150
	3iGS [10]	24.42	0.865	0.184	20.09	0.880	0.145
	GShader [4]	24.28	0.856	0.209	20.65	0.878	0.160
	GShader* [4]	24.89	0.865	0.198	21.27	0.886	0.152
	3DGS-DR [13]	25.37	0.857	0.223	22.02	0.882	0.164
	3DGS-DR* [13]	22.51	0.837	0.239	19.63	0.866	0.180
	Ours	**26.89**	**0.875**	0.183	**22.38**	**0.890**	0.142
VASE (train: 201, test:100) 	3DGS [5]	33.16	0.944	0.093	26.42	0.975	0.042
	GOF [14]	**33.28**	0.948	0.083	26.36	0.975	0.040
	GOF* [14]	32.72	0.945	0.087	25.32	0.973	0.045
	3iGS [10]	33.02	0.943	0.091	26.60	0.975	0.042
	GShader [4]	30.33	0.912	0.129	25.14	0.973	0.046
	GShader* [4]	30.79	0.919	0.121	25.44	0.973	0.046
	3DGS-DR [13]	31.35	0.914	0.149	25.86	0.973	0.046
	3DGS-DR* [13]	30.94	0.912	0.152	25.10	0.971	0.049
	Ours	33.14	**0.949**	**0.076**	**27.20**	**0.982**	**0.027**
POT (train: 201, test:100) 	3DGS [5]	29.88	0.923	0.096	23.50	0.945	0.062
	GOF [14]	29.71	0.921	0.093	23.41	0.943	0.058
	GOF* [14]	29.04	0.919	0.105	22.22	0.940	0.071
	3iGS [10]	**31.13**	0.928	0.090	23.88	0.947	0.059
	GShader [4]	29.37	0.913	0.116	22.53	0.942	0.068
	GShader* [4]	29.53	0.915	0.114	22.86	0.943	0.066
	3DGS-DR [13]	30.22	0.917	0.115	23.34	0.944	0.066
	3DGS-DR* [13]	28.87	0.909	0.125	22.23	0.940	0.074
	Ours	30.30	**0.933**	**0.075**	**24.79**	**0.959**	**0.035**

Table 5. Quantitative comparison results with state-of-the-art methods on each of the 2 synthetic scenes (TOASTER and MIRROR) and 2 real-world scenes (REALBOWL, and REALPOT). We show example images and the dataset splits of each scene in the leftmost column. We report the scores of PSNR, SSIM [11], and LPIPS [15] for entire images and within reflective regions. We mark the **best** and the second best results in each column. ↑ (↓) means higher (lower) is better.

Scene	Method	Entire Image			Reflective Region		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TOASTER (train: 201, test:100) 	3DGS [5]	26.26	0.914	0.123	18.77	0.951	0.060
	GOF [14]	26.34	**0.924**	**0.103**	18.74	0.951	0.060
	GOF* [14]	25.61	0.918	0.115	17.99	0.945	0.070
	3iGS [10]	26.71	0.914	0.120	19.34	0.951	**0.056**
	GShader [4]	25.51	0.897	0.146	18.16	0.944	0.070
	GShader* [4]	26.07	0.900	0.143	18.77	0.946	0.068
	3DGS-DR [13]	25.68	0.885	0.175	18.42	0.949	0.064
	3DGS-DR* [13]	25.97	0.871	0.199	18.95	0.947	0.069
	Ours	**27.73**	0.918	0.115	**20.39**	**0.953**	0.062
MIRROR (train: 201, test:100) 	3DGS [5]	26.65	0.938	0.120	18.65	0.963	0.072
	GOF [14]	26.65	**0.941**	0.112	18.37	0.962	0.074
	GOF* [14]	26.74	**0.941**	0.113	18.52	0.963	0.074
	3iGS [10]	27.86	0.939	0.115	19.90	**0.964**	0.067
	GShader [4]	24.61	0.913	0.156	17.52	0.961	0.075
	GShader* [4]	25.02	0.917	0.149	17.71	0.962	0.074
	3DGS-DR [13]	26.77	0.924	0.145	19.04	**0.964**	0.069
	3DGS-DR* [13]	26.40	0.920	0.152	18.83	0.963	0.075
	Ours	**28.64**	0.938	**0.097**	**20.66**	0.963	**0.056**
REALBOWL (train: 120, test:18) 	3DGS [5]	25.75	0.832	0.212	20.73	0.967	0.041
	GOF [14]	25.79	0.835	0.202	20.71	0.966	0.039
	GOF* [14]	25.59	0.834	0.205	19.81	0.965	0.043
	3iGS [10]	25.34	0.819	0.216	20.80	0.967	0.040
	GShader [4]	24.76	0.817	0.240	19.68	0.966	0.045
	GShader* [4]	24.82	0.819	0.239	19.82	0.966	0.045
	3DGS-DR [13]	25.66	0.832	0.227	20.58	0.967	0.042
	3DGS-DR* [13]	25.48	0.830	0.236	19.86	0.966	0.045
	Ours	**26.16**	**0.839**	**0.195**	**22.84**	**0.973**	**0.026**
REALPOT (train: 121, test:18) 	3DGS [5]	23.89	0.814	0.245	21.89	**0.962**	0.056
	GOF [14]	23.94	**0.817**	0.235	21.78	0.960	0.055
	GOF* [14]	23.80	0.816	0.240	21.07	0.959	0.059
	3iGS [10]	23.40	0.799	0.249	21.63	0.960	0.055
	GShader [4]	23.11	0.802	0.271	20.77	0.960	0.061
	GShader* [4]	23.23	0.806	0.267	20.89	0.961	0.060
	3DGS-DR [13]	23.91	0.816	0.258	21.78	**0.962**	0.058
	3DGS-DR* [13]	23.71	0.815	0.266	20.90	0.961	0.064
	Ours	**24.06**	0.816	**0.230**	**22.69**	**0.962**	**0.044**

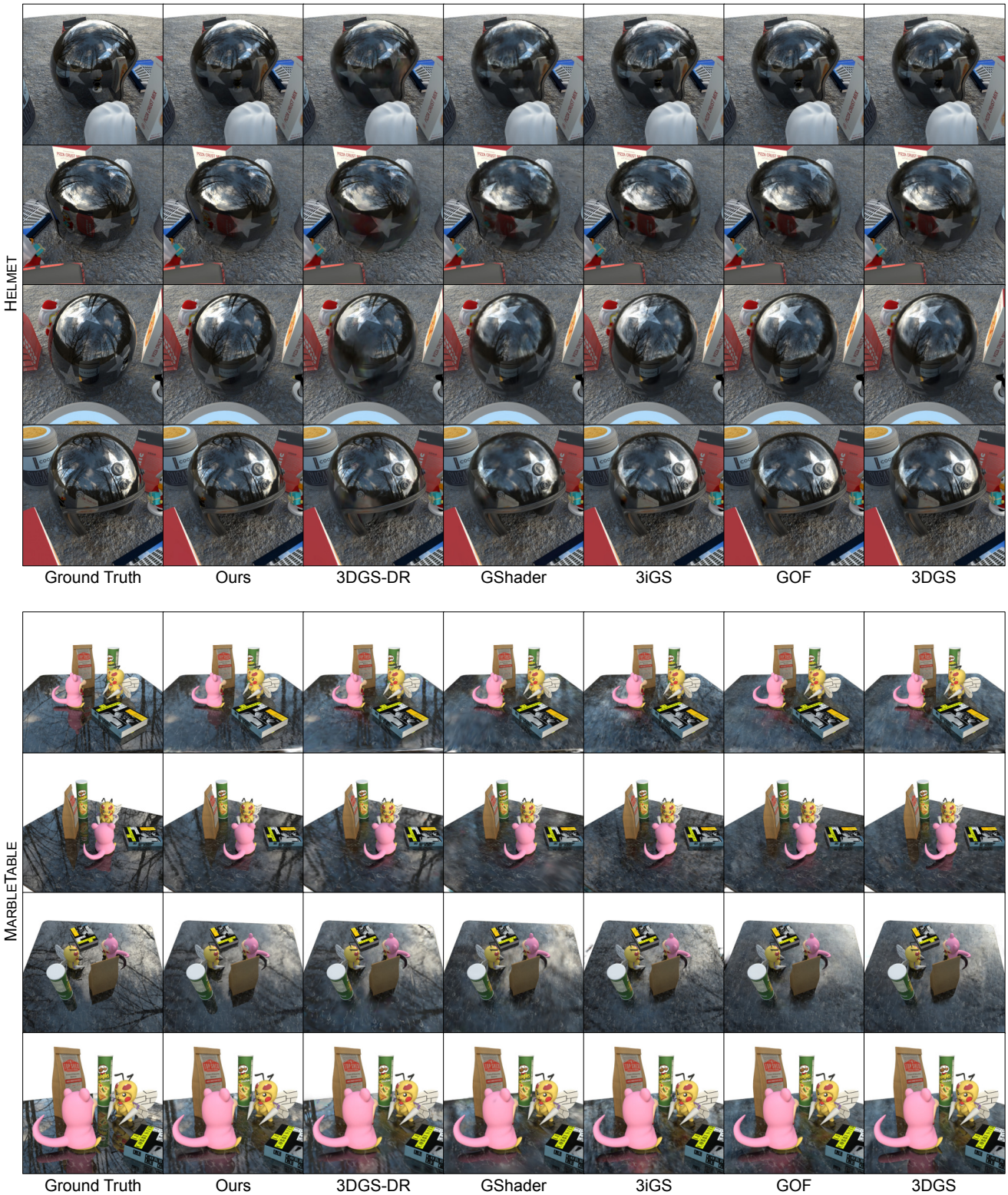

Figure 11. Comparison with state-of-the-art methods on two synthetic scenes: HELMET and MARBLETABLE.

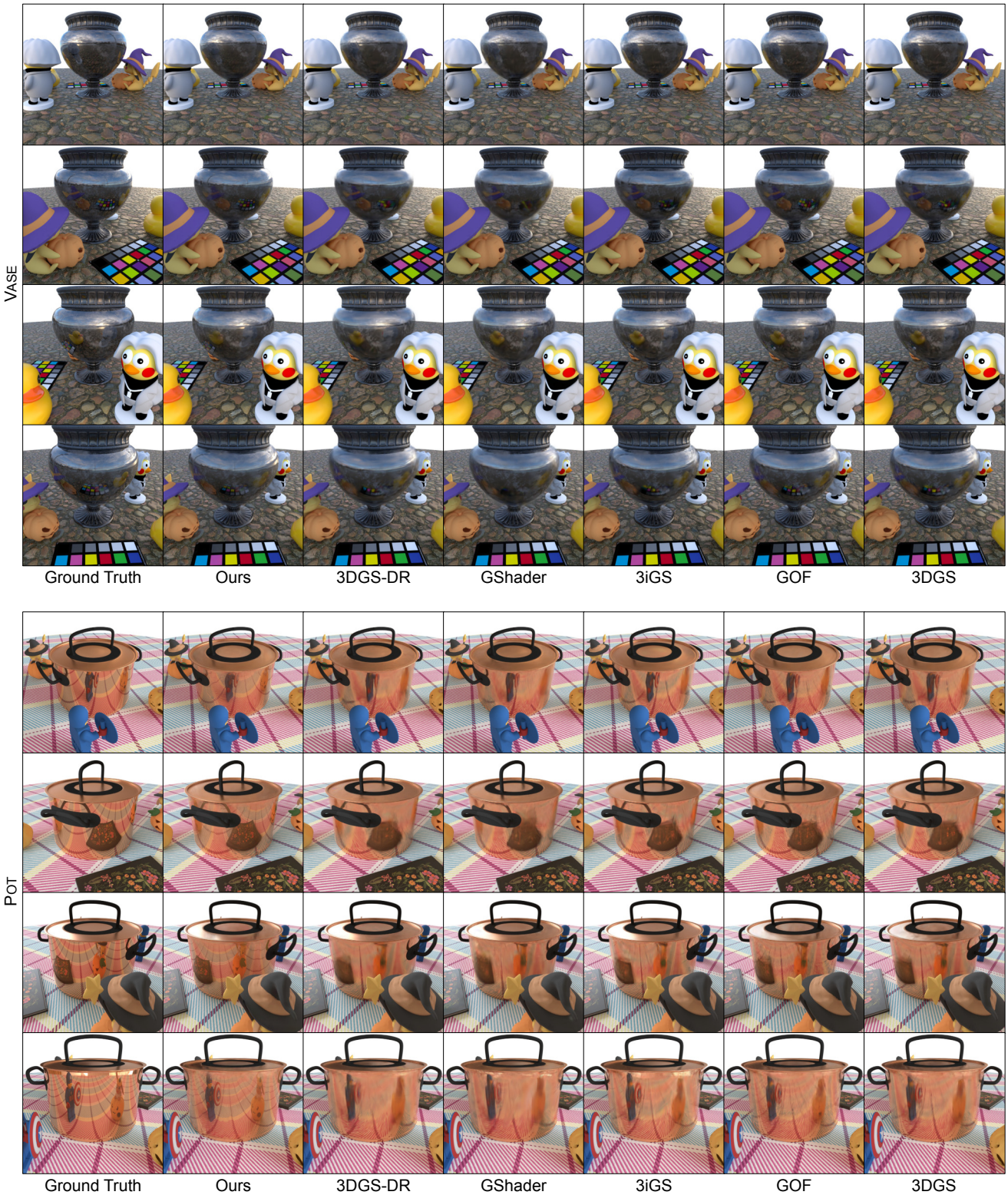

Figure 12. Comparison with state-of-the-art methods on two synthetic scenes: VASE and POT.

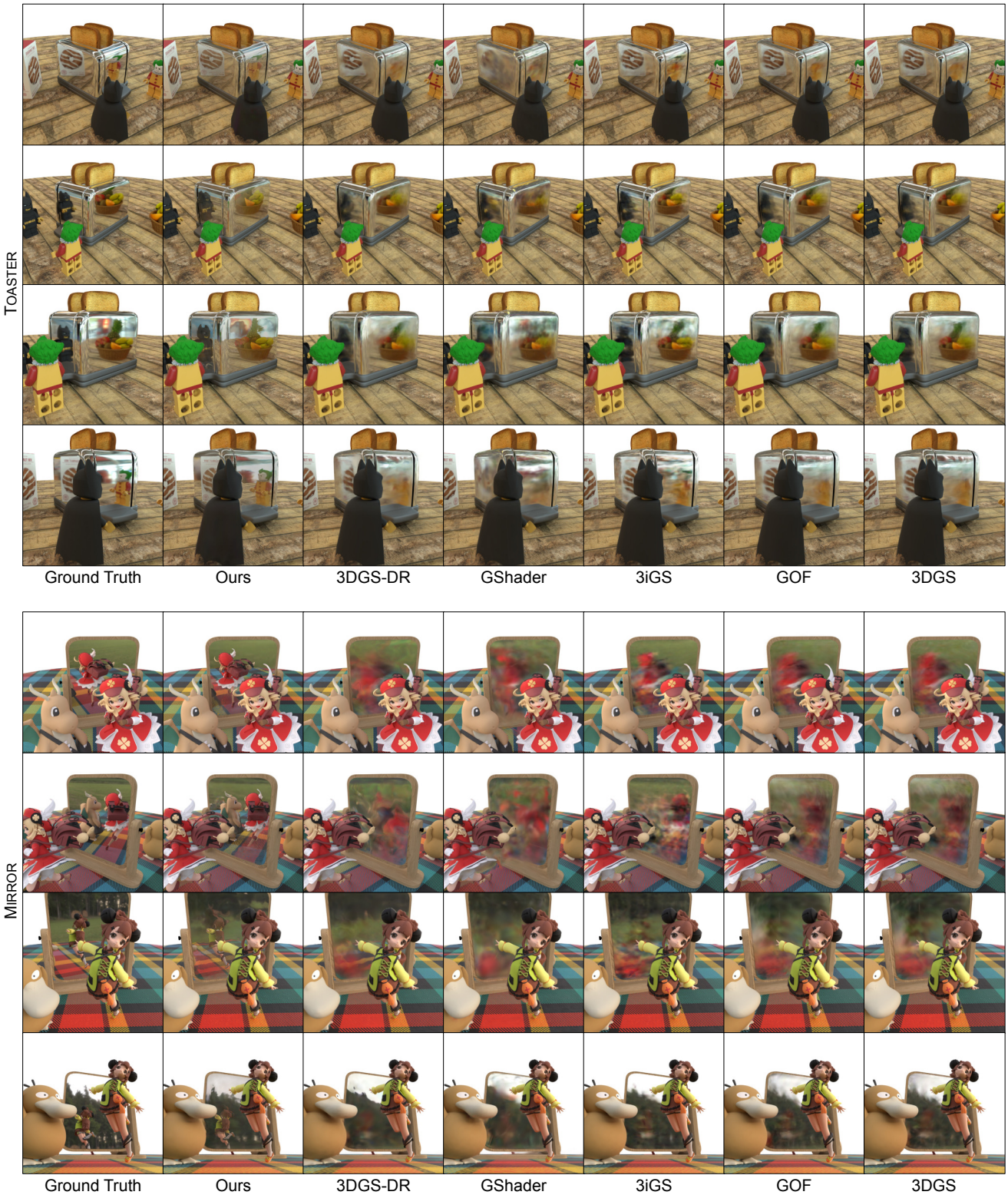

Figure 13. Comparison with state-of-the-art methods on two synthetic scenes: TOASTER and MIRROR.

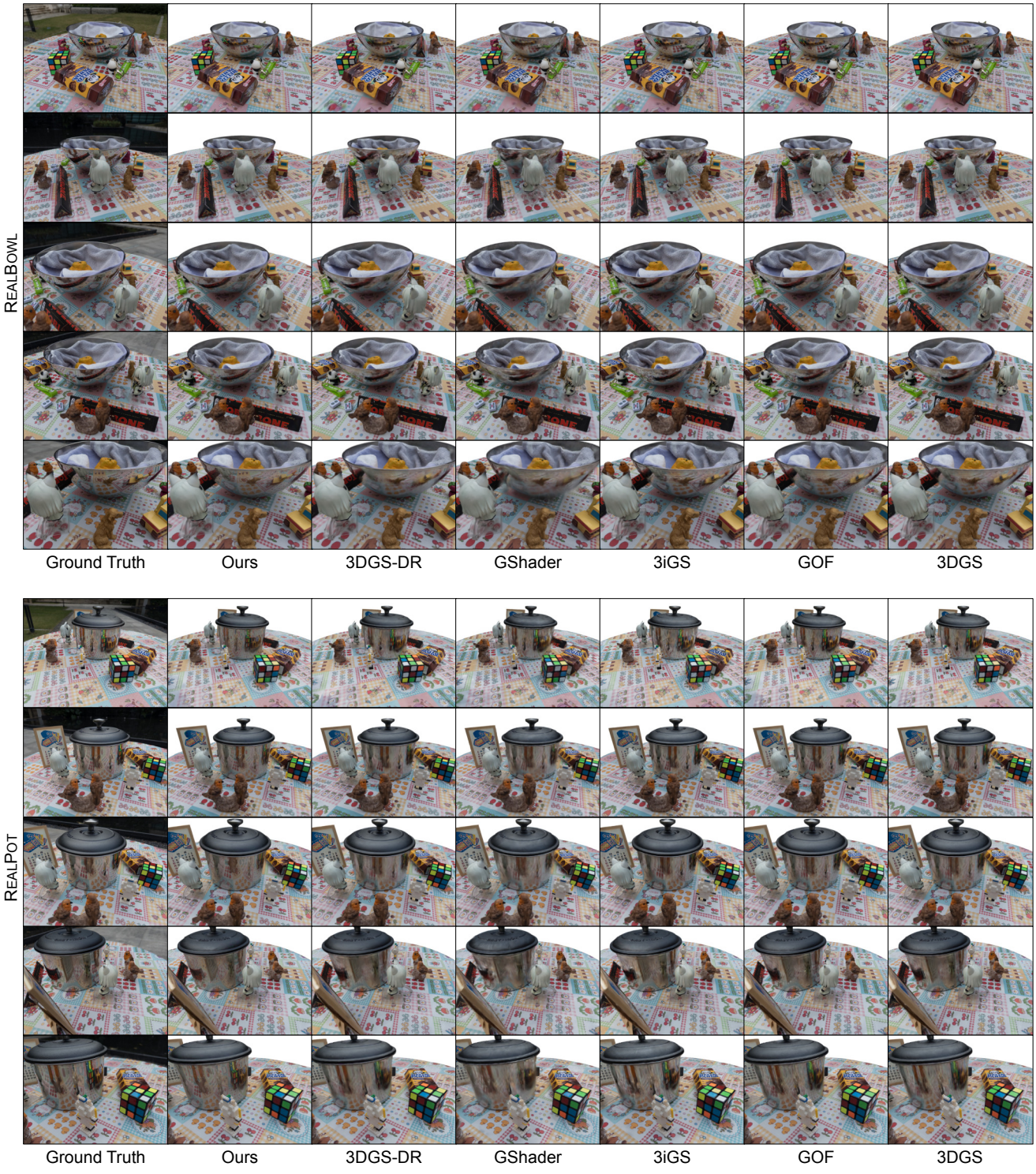

Figure 14. Comparison with state-of-the-art methods on two real-world scenes: REALBOWL and REALPOT.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 13

[2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018. 12

[3] Cycles Developers. Cycles: Open source production rendering, 2024. https://www.cycles-renderer.org/. Accessed: 2024-11-06. 12

[4] 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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 14, 15, 16

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):139:1–139:14, 2023. 12, 14, 15, 16

[6] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local feature matching at light speed. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 12

[7] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David P. Luebke, David K. McAllister, Morgan McGuire, R. Keith Morley, Austin Robison, and Martin Stich. OptiX: a general purpose ray tracing engine. In *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 2010. 11

[8] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 12

[9] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. of European Conference on Computer Vision (ECCV)*, 2016. 12

[10] Zhe Jun Tang and Tat-Jen Cham. 3iGS: Factorised tensorial illumination for 3D Gaussian splatting. In *Proc. of European Conference on Computer Vision (ECCV)*, 2024. 14, 15, 16

[11] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004. 15, 16

[12] Chongjie Ye, Lingteng Qiu, Xiaodong Gu, Qi Zuo, Yushuang Wu, Zilong Dong, Liefeng Bo, Yuliang Xiu, and Xiaoguang Han. StableNormal: Reducing diffusion variance for stable and sharp normal. In *Proc. of the ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia (SIGGRAPH Asia)*, 2024. 12

[13] Keyang Ye, Qiming Hou, and Kun Zhou. 3D Gaussian splatting with deferred reflection. In *Proc. of the ACM SIGGRAPH Conference and Exhibition On Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2024. 14, 15, 16

[14] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):271:1–271:13, 2024. 11, 12, 14, 15, 16

[15] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 15, 16