

# 3D Gaussian Splatting with Deferred Reflection

Keyyang Ye  
State Key Lab of CAD&CG  
Zhejiang University  
Hangzhou, China  
yekeyang@zju.edu.cn

Qiming Hou  
State Key Lab of CAD&CG  
Zhejiang University  
Hangzhou, China  
hqm03ster@gmail.com

Kun Zhou\*  
State Key Lab of CAD&CG  
Zhejiang University  
Hangzhou, China  
kunzhou@acm.org



**Figure 1: Novel view synthesis with specular reflection.** From left to right: Ref-NeRF [Verbin et al. 2022], Neural Point Catacaustics [Kopanas et al. 2022], 3D Gaussian Splatting [Kerbl et al. 2023], GaussianShader [Jiang et al. 2023], our method, ground-truth. We render high quality reflection at a speed comparable with the original reflection-oblivious 3D Gaussian Splatting. The key contribution is a *deferred shading* pipeline, which offers high-precision shading in real-time and enables gradual propagation of correct normal estimation.

## ABSTRACT

The advent of neural and Gaussian-based radiance field methods have achieved great success in the field of novel view synthesis. However, specular reflection remains non-trivial, as the high frequency radiance field is notoriously difficult to fit stably and accurately. We present a deferred shading method to effectively render specular reflection with Gaussian splatting. The key challenge comes from the environment map reflection model, which requires accurate surface normal while simultaneously bottlenecks normal estimation with discontinuous gradients. We leverage the per-pixel reflection gradients generated by deferred shading to bridge the optimization process of neighboring Gaussians, allowing nearly correct normal estimations to gradually propagate and eventually spread over all reflective objects. Our method significantly outperforms state-of-the-art techniques and concurrent work in synthesizing high-quality specular reflection effects, demonstrating a consistent improvement of peak signal-to-noise ratio (PSNR) for both synthetic and real-world scenes, while running at a frame rate almost identical to vanilla Gaussian splatting.

\*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA  
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 979-8-4007-0525-0/24/07  
<https://doi.org/10.1145/3641519.3657456>

## CCS CONCEPTS

- Computing methodologies → Rasterization; Point-based models; Machine learning approaches; Rendering.

## KEYWORDS

Novel view synthesis, deferred shading, real-time rendering

### ACM Reference Format:

Keyyang Ye, Qiming Hou, and Kun Zhou. 2024. 3D Gaussian Splatting with Deferred Reflection. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24)*, July 27-August 1, 2024, Denver, CO, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3641519.3657456>

## 1 INTRODUCTION

Novel view synthesis of 3D scenes captured with multiple images has been a long-standing research topic in computer graphics and vision. Recently, Neural Radiance Field (NeRF) methods first introduced by Mildenhall et al. [2020] have gained popularity by using volume rendering with implicit fields via Multi Layer Perceptrons (MLPs), achieving state-of-the-art visual quality. More recently, 3D Gaussian Splatting [Kerbl et al. 2023] (3DGS) offers a more compelling solution by modeling radiance fields as sparsely distributed 3D Gaussians, synthesizing novel views at high resolution and real-time frame rates, while maintaining state-of-the-art visual quality and competitive training.

However, specular reflection remains challenging for Gaussian splatting to model. Although 3DGS provides view-dependent coloring via per-Gaussian SH (Spherical Harmonics) functions, its directional frequency is too limited to model specular reflection. The training process instead hallucinates Gaussians to explicitly

model the reflected image, which lacks a well-defined spatial position for non-planar reflectors. As such, specular effects end up poorly emulated, with a side effect of compromising geometry quality.

In this paper, we introduce a deferred shading method to effectively render specular reflection with Gaussian splatting. Our method associates each Gaussian with a scalar parameter of reflection strength and regards the shortest axis of each Gaussian ellipsoid as its normal vector. The rendering is performed in two passes. First, a Gaussian splatting pass generates several screen-space maps of base color, normal, and reflection strength. Second, a pixel shading pass queries an environment map with the reflection direction to acquire the specular reflection color, and renders the final color as the sum of the basic and reflection colors weighted by the reflection strength. The environment map, per-Gaussian reflection strength, as well as other Gaussian parameters are all learned during training.

The seemingly mundane environment map query presents significant challenge to the training procedure. As a high frequency lookup table, the environment map places a high precision demand on the normal vectors required to compute reflection directions, while scarcely providing any useful gradient to refine them. On top of that, we only have a semi-transparent Gaussian soup with loosely-defined surface. To this end, we present a training algorithm featuring *normal propagation*. Specifically, based on the observation that Gaussians with relative large reflective strength values have near-correct normal vectors, we expand these reflective Gaussians to propagate their normal vectors to nearby Gaussians. In this way, after one Gaussian with near-correct normal overlaps a different Gaussian without one, some shared pixels can also have near-correct normal, which will get meaningful normal gradients, helping to optimize the normal of the later Gaussian.

Our deferred shading model is critical to the efficacy of training. The Gaussian splatting pass blends Gaussian properties like base color and normal into viewport-aligned textures. The blended input values on each pixel are used to evaluate reflection and compose the base and reflection colors into the final color, which feeds gradient back to input values of the same pixel through image color loss. This creates a gradient channel from color to blended normal to individual Gaussian normal, facilitating information flow between the normal of different Gaussians overlapping the same pixel, which enables normal propagation. This is not possible with per-Gaussian shading, where gradients propagate from color to individual Gaussian normal directly and different Gaussians get independent normal gradients that cannot influence each other.

Experimental results on several previously published datasets show that our method significantly outperforms state-of-the-art methods and concurrent work in synthesizing high-quality specular reflection effects (see Fig. 1), demonstrating a consistent improvement of peak signal-to-noise ratio (PSNR) for both synthetic and real-world scenes, while running at real-time frame rates almost identical to vanilla Gaussian splatting. Our method also produces more accurate normal and environment map estimation (see Fig. 11 and Fig. 12). On the other hand, it does not provide a full geometry-lighting-material decomposition for inverse rendering or relighting. Our shading model separately handles mirror reflection and leaves

rough reflection, anisotropic / layered materials, and global illumination effects to the base SH colors, achieving high-quality rendering superior to full-decomposition methods in novel view synthesis.

## 2 RELATED WORK

Research on novel view synthesis has a long history in computer graphics and vision, and is still developing rapidly. In this section we only review the most relevant references. Please refer to [Tewari et al. 2022; Xie et al. 2022] for comprehensive reviews of the field.

*Novel View Synthesis.* A variety of methods have been proposed to synthesize novel views from multiple images of a static scene [Davis et al. 2012; Gortler et al. 1996; Levoy and Hanrahan 1996]. Recently, Neural Radiance Field (NeRF) [Mildenhall et al. 2020] has gained popularity. NeRF represents the scene as implicit fields of view-dependent color and density, typically evaluated as a deep MLP, and synthesizes high-quality images through volumetric ray marching.

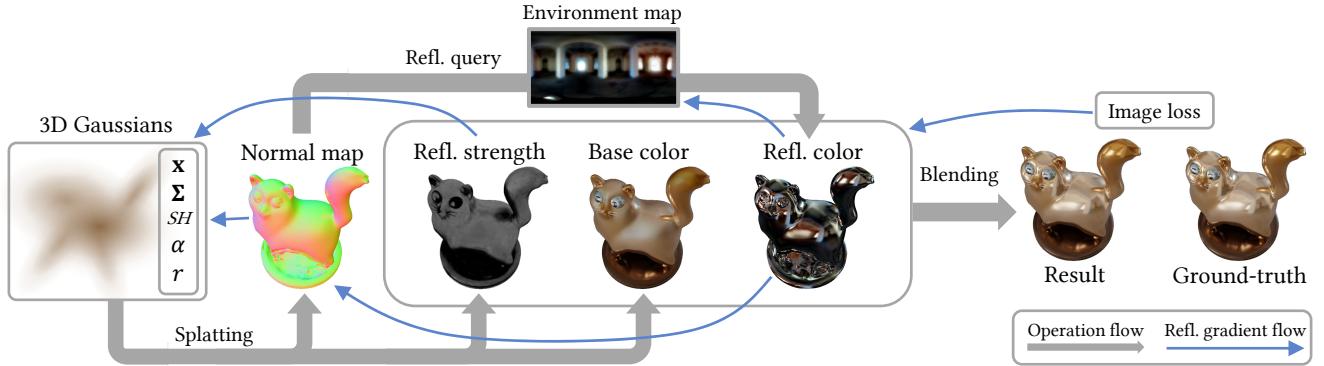
Many follow-up research on NeRF try to improve the image quality (e.g., [Barron et al. 2022; Verbin et al. 2022]), training/rendering performance (e.g., [Garbin et al. 2021; Müller et al. 2022; Yu et al. 2021]), sparse-view generalization ability (e.g., [Chen et al. 2021; Deng et al. 2022; Jain et al. 2021]). One notable work [Müller et al. 2022] replaces the deep MLP of classical NeRF with a shallow MLP featuring multi-resolution hash encoding as input, which enables rapid training and real-time rendering. As a generic neural network design with applications beyond novel view synthesis, it is orthogonal to our approach, which does not use neural networks.

Rendering novel views of reflective objects is challenging as the radiance field becomes high frequency in the view dimension, making it difficult to reconstruct from the spatial images typically used as input. Classical NeRFs assume low frequency view-dependency. Ref-NeRF [Verbin et al. 2022] extends NeRFs with a novel parameterization for view-dependent outgoing radiance and normal vector regularization. The added dimensions are challenging to optimize, though, which makes its training time-consuming and leads to noisy synthesis results.

The radiance field can also be represented by a point cloud [Xu et al. 2022; Zhang et al. 2022a], which enables unique algorithm designs. Kopanas et al. [2022] formalize hallucinated mirror images as catacaustics consisting of virtual points inside the reflector. Curved surfaces are handled using MLPs to adjust the point positions based on camera movement. However, the training of this MLP is typically under-constrained due to static input cameras, resulting in unstable behavior on novel views.

The more recent 3D Gaussian splatting [Kerbl et al. 2023] departs from the volumetric NeRF formulation in favor of a differentiable rasterizer for spatial Gaussians, achieving a distinctive hard-realtime frame rate at 1080p resolutions. Our approach extends its Gaussian representation, significantly enhancing specular reflection effects while maintaining the original training and rendering efficiency.

*Inverse Rendering.* It is tempting to generalize novel view synthesis to *inverse rendering* [Boss et al. 2021; Jiang et al. 2023; Srinivasan et al. 2021; Zhang et al. 2021a,b, 2022b], which aims to completely decompose geometry, lighting and material, typically employing a physically-based shading model. While such a scene model can



**Figure 2: Our rendering pipeline.** A Gaussian splatting pass is first performed to bake reflection strength, normal, and base color to screen space maps. In the following shading pass, for each pixel, we use the normal map to compute a reflection direction and query an environment map for a reflected color. The reflection strength is then used to blend base color and reflection color into the final result. An image loss is used to back-propagate gradients. Note that there exist many gradient propagation paths. Here we only illustrate the gradient flow most relevant to reflection fitting.

automatically render a wide range of visual effects including reflection, its optimization is significantly under-constrained, especially when starting from a few static images. The corresponding optimization tends to rely on prior assumptions about the scene, such as material or geometry smoothness, compromising the overall quality when applied to novel view synthesis.

Recent Signed Distance Field (SDF) methods [Liang et al. 2023; Liu et al. 2023; Munkberg et al. 2022] utilize the NeRF neural field as a geometry representation, leading to an elegant inverse rendering formulation that allows flexible lighting and material changes. The flexibility comes at a cost, as they inherit the high training cost of NeRF and the inherent smoothness of neural SDFs tends to over-smooth geometry details.

The concurrent work of GaussianShader [Jiang et al. 2023] approaches the inverse rendering goal using a Gaussian-based scene representation, estimating physically-based BRDF parameters and a normal vector for each Gaussian. Their method inherits the real-time performance of 3DGs, albeit with noticeable overhead.

Both our method and GaussianShader tackle the same specular reflection problem in a Gaussian splatting setting. The key difference is that we compute reflection on pixels as opposed to Gaussians, generating more reflection samples for the same rendering cost. The extra samples stabilize reflection strength and environment map training by smoothing out gradient, leading to significantly less noise in the final result. Our per-pixel shading also eliminates interpolation artifacts caused by discontinuous color change at Gaussian boundaries.

**Deferred Shading.** Deferred shading [Deering et al. 1988] is a classical real-time rendering technique that computes high frequency shading effects like specularity per-pixel in screen space, after baking scene properties like positions and normal into viewport-aligned textures. Representing a scene as a neural texture map on top of 3D meshes, the deferred neural rendering approach [Thies et al. 2019] first rasterizes the scene into a screen-space feature map, which is then converted to photo-realistic images based on a

neural network. Both the neural network and the neural texture are trained end-to-end. Hedman et al. [2021] propose a deferred NeRF architecture, which renders screen-space diffuse color and feature vector maps using a deep MLP, followed by another shallow MLP to predict the view-dependent specular color for each pixel. We combine deferred shading with 3D Gaussian splatting to effectively render specular reflection, which also produces accurate estimation of normal and environment maps.

## 3 METHOD

### 3.1 Rendering Model

Our deferred rendering model consists of two passes. The first is Gaussian splatting. Following the original setting of the 3DGs renderer [Kerbl et al. 2023], we start with Gaussian parameters  $\Theta_i$ , per-Gaussian view-dependent SH colors  $c_i(\mathbf{v})$ , and compute the pixel colors  $C(\mathbf{v})$ . Here  $i$  is the Gaussian index and  $\mathbf{v}$  refers to the view direction. For simplicity, we also parameterize the output image with  $\mathbf{v}$ , and treat the splatting process as a blackbox that blends colors with linear weights  $G$ :

$$C(\mathbf{v}) = \sum_i c_i(\mathbf{v}) G(\Theta_i, \mathbf{v}). \quad (1)$$

Here the most expensive component is the weights  $G$ . It is computed at the finest per-Gaussian-per-pixel granularity, is order-sensitive and requires a sort as a preprocess. With  $G$  already computed, though, it is very cheap to blend extra per-Gaussian values alongside  $c_i$ . We apply this to  $n_i$ , the shortest axis of each Gaussian ellipsoid interpreted as its normal vector, and  $r_i$ , a per-Gaussian scalar controlling its specular reflection strength:

$$N(\mathbf{v}) = \sum_i n_i G(\Theta_i, \mathbf{v}), \quad R(\mathbf{v}) = \sum_i r_i G(\Theta_i, \mathbf{v}), \quad (2)$$

where  $n_i$  are flipped as necessary to face the camera.

Second, a deferred reflection pass composes the final pixel color  $C'(\mathbf{v})$ , detached from Gaussians  $G$ :

$$C'(\mathbf{v}) = (1 - R(\mathbf{v}))C(\mathbf{v}) + R(\mathbf{v})E\left(\frac{2\mathbf{v} \cdot N(\mathbf{v})N(\mathbf{v})}{\|N(\mathbf{v})\|} - \mathbf{v}\right), \quad (3)$$

where  $E$  is a learned environment map queried on the reflection direction with a bilinear filter.

Fig. 2 illustrates the components used by our rendering model. The per-Gaussian normal  $n_i$  and reflection strength  $r_i$  are trained and splatted separately. The splatted images are combined in the shading pass to compose the final image. This process works entirely in screen space. The environment map is trained entirely from the final pass, detached from the Gaussians.

### 3.2 Loss Function and Normal Gradient

When training, we use the same combined image loss  $\mathcal{L}_1$  and D-SSIM loss functions as in [Kerbl et al. 2023]:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM}, \quad (4)$$

where  $\lambda = 0.2$  in our implementation.

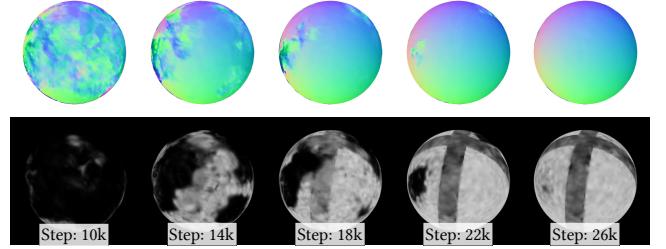
Handling the normal vector gradient  $\frac{\partial \mathcal{L}}{\partial N}$  is a challenge in training. As the loss function is purely color-based, our normal gradient ultimately comes from the environment map:  $\frac{\partial \mathcal{L}}{\partial N} = \frac{\partial \mathcal{L}}{\partial E} \frac{\partial E}{\partial N}$ . With  $E$  being a texture query, the only non-zero component of  $\frac{\partial E}{\partial N}$  comes from the bilinear texture filter. Intuitively, this can be interpreted as the gradient descent process rotating the reflection direction towards the environment map texel best matching the expected pixel color, by updating the underlying  $N$ . However, the rotation target is restricted to the four texels participating in the bilinear texture filter of a particular query, limiting meaningful gradients to normal vectors already close to the correct value.

Fortunately, our deferred shading model provides an elegant solution. Since we perform the environment lookup at the pixel level, each Gaussian only needs to cover a few pixels with near-correct normal to receive meaningful gradients. We leverage this property to propagate correct normals across neighboring Gaussians, eventually expanding to all reflective surfaces we can find. The details will be explained in the next subsection.

### 3.3 Training

We bootstrap our training process with a short view-independent stage by turning off view-dependent color and reflection optimization, where reflection strengths  $r_i$  are initialized to 0 and the per-Gaussian SH color functions  $c_i(\mathbf{v})$  are restricted to order 0, i.e., to constant terms  $c_i(\mathbf{v}) = c_{i,0}$ . With  $r_i = 0$ , reflection-related optimizations are disabled as relevant gradients have zero magnitude. This stage lasts for a few thousand iterations, typically taking a few minutes on a high-end GPU. The optimization process is the same as in the original 3DGS [Kerbl et al. 2023].

In the following training, we turn on the optimization of per-Gaussian reflection strength  $r_i$  and environment map. A few Gaussians may get relatively large reflection strength during optimization (i.e.,  $r_i > 0.1$ ). Observing that such reflective Gaussians have near-correct normal vectors (see Fig. 3), we propose to propagate their normal vectors to nearby Gaussians. Specifically, when one Gaussian with near-correct normal overlaps a different Gaussian without one, some shared pixels can also have near-correct normal,



**Figure 3: The propagation of Gaussian normal and reflection strength at various training steps.**

which will get meaningful normal gradients and help to optimize the normal of the later Gaussian. To facilitate such propagation, we periodically raise the opacity of all Gaussians to at least 0.9 and the reflection strength to at least 0.001, then scale up the two longest axes of reflective Gaussians by 1.5×, leaving the shortest used-as-normal axes intact. This makes almost every reflective Gaussian overlap with its neighbors, and the universally-high opacity ensures that every visible Gaussian contributes a significant magnitude to surface normal and in reciprocal, getting influenced by meaningfully normal gradients during back propagation. We call this process *normal propagation*.

Fig. 3 shows a concrete example of how correct normals propagate as training proceeds. Starting from noisy shortest axis directions generated by constant color fitting, some randomly-correct spots first gained reflection strength by step 9000. They then propagate their correct normal vectors to neighboring Gaussians, gradually spreading over the noisy bumps, and finally yielding a smooth sphere. The reflection strength map increases early but only starts approaching the correct value as normal vectors become more accurate.

The diffuse colors  $c_{i,0}$  have already been optimized by the view-independent bootstrap and during reflection training they tend to over-fit, which can hinder reflective surface discovery. To counter this effect, we intentionally sabotage the colors of not-yet-reflective Gaussians with  $r_i \leq 0.1$ , by adding a ±10% noise whenever normal propagation is applied. We call this process *color sabotage*.

Our periodic opacity raise conflicts with the periodic opacity clamping in original 3D Gaussian Splatting, which clamps opacity to no more than 0.01 [Kerbl et al. 2023]. The color sabotage also prevents the color terms from converging. As a workaround, we interleave opacity-raising periods with opacity-clamping periods so that they are never applied simultaneously. We also terminate normal propagation and color sabotage once the number of Gaussians with  $r_i > 0.1$  stop increasing for a fixed number of iterations, which indicates that no more specular surfaces can be found. We only start to optimize higher-order color SH coefficients after this specular termination criteria has been met, to prevent them from interfering with reflection training.

## 4 RESULTS AND EVALUATION

We conduct comprehensive experiments on a workstation with an i7-13700KF CPU, 32GB memory and an NVIDIA RTX 4090 GPU, to demonstrate the effectiveness and efficiency of our approach.

**Table 1: Per-scene image quality comparison on synthesized test views.**

Datasets		Shiny Blender [Verbin et al. 2022]						Glossy Synthetic [Liu et al. 2023]						Real		
		ball	car	coffee	helmet	teapot	toaster	bell	cat	luyu	potion	tbell	teapot	garden	sedan	toy
PSNR $\uparrow$	Ref-NeRF	33.16	30.44	33.99	29.94	45.12	26.12	30.02	29.76	25.42	30.11	26.91	22.77	22.01	25.21	23.65
	NPC	23.76	24.19	30.39	25.59	41.22	19.76	22.41	25.35	23.68	23.09	19.03	18.21	21.01	24.77	22.84
	3DGS	27.65	27.26	32.30	28.22	45.71	20.99	25.11	31.36	26.97	30.16	23.88	21.51	21.75	26.03	23.78
	GShader	30.99	27.96	32.39	28.32	45.86	26.28	28.07	31.81	27.18	30.09	24.48	23.58	21.74	24.89	23.76
	ENVIDR	41.02	27.81	30.57	32.71	42.62	26.03	30.88	31.04	28.03	32.11	28.64	26.77	21.47	24.61	22.92
	Ours, forward	27.64	28.99	31.61	28.01	45.68	24.83	25.74	32.22	27.01	30.25	24.11	23.13	21.49	26.05	23.49
	Ours, deferred	33.66	30.39	34.65	31.69	47.12	27.02	31.65	33.86	28.71	32.29	28.94	25.36	21.82	26.32	23.83
SSIM $\uparrow$	Ref-NeRF	0.971	0.950	0.972	0.954	0.995	0.921	0.941	0.944	0.901	0.933	0.947	0.897	0.584	0.720	0.633
	NPC	0.908	0.898	0.955	0.938	0.994	0.835	0.892	0.921	0.854	0.877	0.742	0.762	0.558	0.711	0.547
	3DGS	0.937	0.931	0.972	0.951	0.996	0.894	0.908	0.959	0.916	0.938	0.900	0.881	0.571	0.771	0.637
	GShader	0.966	0.932	0.971	0.951	0.996	0.929	0.919	0.961	0.914	0.936	0.898	0.901	0.576	0.728	0.637
	ENVIDR	0.997	0.943	0.962	0.987	0.995	0.922	0.954	0.965	0.931	0.960	0.947	0.957	0.561	0.707	0.549
	Ours, forward	0.939	0.941	0.968	0.947	0.996	0.919	0.909	0.964	0.911	0.938	0.904	0.891	0.566	0.767	0.626
	Ours, deferred	0.979	0.962	0.976	0.971	0.997	0.943	0.962	0.976	0.936	0.957	0.952	0.936	0.581	0.773	0.639
LPIPS $\downarrow$	Ref-NeRF	0.166	0.050	0.082	0.086	0.012	0.083	0.102	0.104	0.098	0.084	0.114	0.098	0.251	0.234	0.231
	NPC	0.237	0.120	0.119	0.156	0.013	0.226	0.203	0.121	0.101	0.174	0.243	0.246	0.302	0.311	0.347
	3DGS	0.162	0.047	0.079	0.081	0.008	0.125	0.104	0.062	0.064	0.093	0.125	0.102	0.248	0.206	0.237
	GShader	0.121	0.044	0.078	0.074	0.007	0.079	0.098	0.056	0.064	0.088	0.122	0.091	0.274	0.259	0.239
	ENVIDR	0.020	0.046	0.083	0.036	0.009	0.081	0.054	0.049	0.059	0.072	0.069	0.041	0.263	0.387	0.345
	Ours, forward	0.156	0.044	0.081	0.082	0.008	0.091	0.104	0.059	0.068	0.096	0.124	0.096	0.252	0.221	0.249
	Ours, deferred	0.098	0.033	0.076	0.049	0.005	0.081	0.046	0.040	0.053	0.075	0.067	0.067	0.247	0.208	0.231

As an ablation study for our core deferred reflection design, we also implement a forward-shading alternative, which computes reflection colors on individual Gaussians and splats them to the final image. The alternative forward renderer is used for both training and testing and other training designs remain unchanged.

**Dataset.** We conduct evaluation on several datasets with specular objects, including two synthetic datasets: Shiny Blender [Verbin et al. 2022] and Glossy Synthetic [Liu et al. 2023], and one real captured dataset used by Ref-NeRF [Verbin et al. 2022]. We also use the non-specular NeRF Synthetic [Mildenhall et al. 2020] dataset as a regression test.

**Implementation details.** For real scenes, we use a spherical domain  $M$  to cover the foreground object. We restrict our deferred reflection stage to Gaussians inside  $M$  to reduce interference from the background during training. Based on our observations, background objects that are only captured in a limited amount of views exhibit a similar behavior as reflective objects, which interferes with our environment map fitting.

**Baselines and metrics.** We compare our method against the following baselines: **3DGS**: vanilla 3D Gaussian Splatting [Kerbl et al. 2023] with no special treatment for reflection; **GShader** [Jiang et al. 2023]: a method that shades each Gaussian with a reflection-aware shader, which can be considered as a differentiable forward rendering pipeline; **ENVIDR** [Liang et al. 2023]: a SDF-based method using neural rendering for inverse rendering; **Ref-NeRF** [Verbin et al. 2022]: a NeRF-based method focusing on reflective objects rendering; **NPC**: Neural Point Catacaustics [Kopanas et al. 2022]: a renderer that warps a point-based hallucinated reflection using MLPs. All methods are applied to the same input data, except for NPC, which requires a mask covering all reflectors for each input image. We use the foreground mask estimated by [Kirillov et al. 2023] as a substitute. We present quantitative results measured with three standard metrics: PSNR, SSIM and LPIPS.

To demonstrate the accuracy normal and light reconstruction, we also compare our method with SDF-based inverse rendering methods: **ENVIDR** [Liang et al. 2023] and **NVDiffRec** [Munkberg et al. 2022]. We use Mean Angular Error in degrees ( $MAE^\circ$ ) to evaluate the normal reconstruction accuracy. We evaluate environment map reconstruction accuracy with LPIPS to mitigate the fundamental ambiguity between albedo, roughness and light.

## 4.1 Comparisons with baselines

**Image quality.** Table 1 presents the quantitative comparison results on three datasets. Our method demonstrates a clear advantage in terms of image quality on synthetic datasets and also shows comparable results on real datasets. The visual comparisons on synthetic and real datasets are shown in Fig. 8, Fig. 9 and Fig. 10. Ref-NeRF [Verbin et al. 2022] shows some regular patterns in the toaster scene. The results from NPC [Kopanas et al. 2022] are noisy and tend to exhibit a diffuse texture. The MLP-predicted warping field is severely strained when the camera viewpoint has a high

**Table 2: Normal and light reconstruction quality (evaluated by MAE° and LPIPS respectively) comparisons on the Shiny Blender Dataset.**

	GShader	NVDiffRec	ENVIDR	Ours
MAE° ↓	22.31	17.02	4.618	4.871
LPIPS ↓	0.621	0.636	0.615	0.511

**Table 3: Training time and rendering frame rates.**

	Shiny Blender		Real scenes	
	Training time	FPS	Training time	FPS
Ref-NeRF	19h	0.06	27h	0.02
NPC	12h	4	23h	1
ENVIDR	3.2h	3	4.7h	1
3DGS	6min	277	17min	84
GShader	60min	51	72min	31
Ours	16min	251	47min	80

**Table 4: Number of Gaussians in the ball, car, helmet and toaster scenes of the Shiny Blender Dataset.**

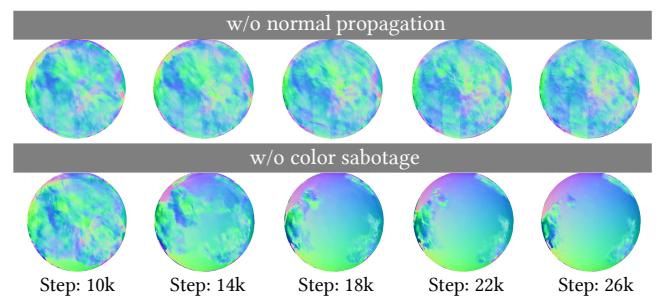
	ball	car	helmet	toaster
Ours, deferred	27.2k	117k	36k	100k
Ours, forward	76.4k	205k	62k	236k
GShader	76.5k	316k	106k	336k
3DGS	199k	307k	100k	342k

degree of freedom and the reflector contains both diffuse and specular components. The results from 3DGS [Kerbl et al. 2023] and GaussianShader [Jiang et al. 2023] are blurred on reflection surfaces or show incorrect reflections. The results from ENVIDR [Liang et al. 2023] over-smooth some local details. Our method produces much smoother-looking surfaces without blurring the reflection details, and stays closer to the ground truth. In close-up views, our result exhibits significantly less visible Gaussian boundaries, typically manifesting as oval-shaped color blobs.

*Normal reconstruction.* Behind the scenes, the quality improvement is backed by improved fitting quality for surface normal and environment map. Table 2 lists the dataset-averaged mean angular error of normal maps on the Shiny Blender Dataset. Despite our method only focusing on the normal of specular surfaces, we still achieve quality comparable to ENVIDR [Liang et al. 2023]. Fig. 11 compares the normal vectors predicted by our method, ENVIDR, Ref-NeRF [Verbin et al. 2022], and GaussianShader [Jiang et al. 2023], alongside ground truth and the 3DGS shortest-axis initialization. As illustrated, our method creates smooth results while preserving sharp boundaries from color loss terms alone, like the sharp separation between the saucer and the cup in the coffee scene. ENVIDR, based on the SDF representation, provides nearly perfect normals for the sphere but fails on detailed geometry, such as car wheel rims and thin cup walls, due to the smoothness of SDF. Ref-NeRF produces sharp, yet noisy results. GaussianShader filters out the noise with a smoothness prior, which unfortunately also blurs out object boundaries like between the saucer and the cup in the coffee scene, and shows some surface defects on the ball.



**Figure 4: Quality comparison between forward and deferred designs. From left to right: teapot, bell, tbell.**



**Figure 5: Normal maps at various training steps with one algorithm component disabled.**

*Light reconstruction.* We quantitatively compare the quality of light reconstruction in Table 2. We achieve the best quality thanks to our simple rendering model, which reduces the ill-posedness of the inverse problem. We also visualize the environment maps reconstructed by GaussianShader, ENVIDR, NVDiffRec, and our method. To compensate for the fundamental light-albedo ambiguity, we equalize the total energy of the two environment maps before comparison. As illustrated in Fig. 12, the per-pixel reflection allows our method to reconstruct a significantly less noisy environment map with almost full directional coverage, whereas GaussianShader can only reconstruct a few texels for each Gaussian, leading to a noisy image with many holes. For SDF-based methods (ENVIDR and NVDiffRec), they struggle to perfectly decompose lighting, geometry, and material, leading to non-robust lighting estimation.

*Efficiency.* Table 3 lists the dataset-averaged training time and rendering frame rate of all tested methods. Frame rate values are computed as the reciprocal of averaged frame render time to reduce the impact of unfairly large numbers on simple scenes. As a reflection-oblivious reference, 3DGS [Kerbl et al. 2023] serves as the performance upper-bound. The NeRF-based method Ref-NeRF [Verbin et al. 2022], SDF-based method ENVIDR [Liang et al. 2023], and point-based method NPC [Kopanas et al. 2022] require dozens of hours to train. GaussianShader [Jiang et al. 2023] requires

**Table 5: The fitting quality impact of disabling various algorithm components.**

Ablations	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Ours	33.66	0.979	0.098
w/o propagation	27.85	0.938	0.159
w/o sabotage	30.00	0.959	0.128

**Table 6: Regression test on non-specular scenes.**

	NeRF Synthetic					
	drums	ficus	hotdog	lego	mic	ship
PSNR $\uparrow$						
3DGS	25.10	28.14	35.52	32.94	31.55	29.06
Ours	25.31	28.03	35.58	32.94	31.97	29.07
SSIM $\uparrow$						
3DGS	0.947	0.965	0.983	0.979	0.986	0.897
Ours	0.946	0.963	0.982	0.978	0.987	0.894
LPIPS $\downarrow$						
3DGS	0.055	0.540	0.032	0.025	0.028	0.124
Ours	0.055	0.055	0.033	0.026	0.028	0.129

an hour to train and renders several times slower than vanilla 3DGS. Our method trains to convergence in under an hour and renders at frame rates almost identical to the 3DGS upper bound. We also compare the final number of Gaussians generated by each method in Table 4. By deferring shading to pixels, our method no longer needs to split Gaussians to meet the shading frequency demand. We are able to fit the same scene with significantly less Gaussians, which also improves our real-time performance.

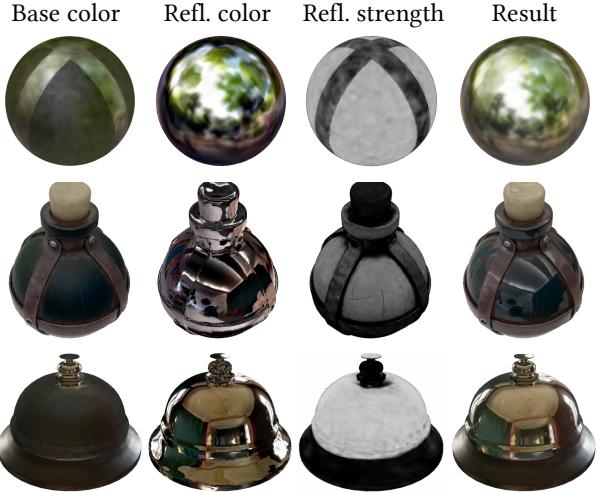
## 4.2 Ablation Study

We conduct various ablation studies to validate the impact of key design choices. Specifically, we compare deferred with forward shading, assess the necessity of normal propagation and color sabotage. We also perform a regression test on the non-specular NeRF Synthetic Dataset [Mildenhall et al. 2020].

*Deferred shading versus forward shading.* As shown in Table 1, using deferred shading in our method results in better PSNR, SSIM and LPIPS, compared to forward shading in all scenes. As illustrated in Fig. 4, deferred shading produces sharp reflections with pixel-level details like the window frames in the rightmost column. The leftmost teapot scene also demonstrates a more complete reconstruction of the reflected environment, producing precise curtain shapes while the forward pipeline fails to capture. We summarize the merits of deferred shading for inverse rendering as two-fold:

- The pixel-precise normal maps alleviate the impact of imprecise Gaussian normal, while also allowing Gaussians with correct normal to propagate gradients to incorrect ones.
- The environment map is more closely linked to the image loss, which accelerates its optimization and in turn helps guiding normal vectors towards the correct direction.

*Necessity of normal propagation and color sabotage.* With the respective algorithm component disabled, Fig. 5 shows the normal maps at various training steps. Table 5 evaluates the quantitative



**Figure 6: Decomposition results of our method. From top to bottom: ball, potion, tbell.**



**Figure 7: Limitations. Top row: inconsistent treatment of car windows. Bottom row: slow convergence on concave bell.**

image metrics after convergence. Without normal propagation, the normal map stays almost unchanged for the entire training. Without color sabotage, normal propagates significantly slower and converges prematurely, outpaced by overfit Gaussian colors. Both result in a significant drop of final image quality.

*Decomposition results.* We visualize the base color map, reflection color map, reflection strength map and final results in Fig. 6. Our method precisely captures the specular reflections while leaving other effects to base SH colors, like the rough bands in the sphere scene and the glossy rim of the tbell scene.

*Regression in non-specular scene.* Table 6 compares our method with plain 3DGS [Kerbl et al. 2023] on non-specular scenes. We are able to achieve an almost indistinguishable quality with minimal regression. As we use the same image loss function with no extra regularization, we are able to correctly estimate the reflection strength as zero for diffuse objects. While our normal propagation depends on reflection, the SH shading we retain does not use normal at all and normal quality on diffuse objects is inconsequential.

## 4.3 Limitations

Our method can handle at most one layer of reflective materials per pixel, which is inherited from traditional deferred shading. The car

windows in the top row of Fig. 7 show two different behaviors of our algorithm on transparent objects. The front window converges to a purely reflective surface and the side windows converges to purely transparent surfaces. A fundamental solution would first require a way to reliably create Gaussians on the transparent windows during initialization.

Normal propagation works less efficient on concave scenes, like the bell scene in the bottom row of Fig. 7. Training still converges but takes considerably more time. A better designed optimization strategy can potentially fix this issue.

## 5 CONCLUSION

We have presented a high-quality deferred Gaussian splatting renderer specializing in reflection. It demonstrates stable training and competitive visual quality at almost identical frame rates to vanilla 3D Gaussian splatting, also producing accurate surface normal and environment maps.

Our deferred shading approach may open up many possibilities for future exploration. It would be interesting to explore more creative splits of the rendering equation in the context of Gaussian splatting. Our pipeline can also be extended to higher quality reflection algorithms beyond an environment map, including screen-space reflections [McGuire and Mara 2014] and hardware ray tracing. Generalizing 3D Gaussians and differentiable rendering to such methods can lead to significantly better reflection qualities. It is also interesting to explore the possibility of adding a physically-based roughness, generalizing our method to glossy materials.

## ACKNOWLEDGMENTS

This work is partially supported by NSF China (No. 62227806 & U23A20311) and the XPLORER PRIZE. The source code and data are available at <https://github.com/gapszju/3DGS-DR>.

## REFERENCES

- Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF CVPR*. 5470–5479.
- Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. 2021. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF CVPR*. 12684–12694.
- Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. 2021. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE CVPR*. 14124–14133.
- Abe Davis, Marc Levoy, and Fredo Durand. 2012. Unstructured light fields. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 305–314.
- Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. 1988. The triangle processor and normal vector shader: a VLSI system for high performance graphics. In *Proceedings of SIGGRAPH '88*. ACM, New York, NY, USA, 21–30.
- Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF CVPR*. 12882–12891.
- Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14346–14355.
- Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. In *Proceedings of SIGGRAPH '96*. ACM, New York, NY, USA, 43–54.
- Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. In *Proceedings of the IEEE/CVF ICCV*. 5875–5884.
- Ajay Jain, Matthew Tancik, and Pieter Abbeel. 2021. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. In *Proceedings of the IEEE CVPR*. 5885–5894.
- Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuxin Ma. 2023. GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces. *arXiv:2311.17977 [cs.CV]*
- Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (July 2023).
- Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643* (2023).
- Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. 2022. Neural Point Catacaustics for Novel-View Synthesis of Reflections. *ACM Trans. Graph.* 41, 6, Article 201 (nov 2022), 15 pages.
- Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proceedings of SIGGRAPH '96*. Association for Computing Machinery, New York, NY, USA, 31–42.
- Ruofan Liang, Huiting Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. 2023. Envindr: Implicit differentiable renderer with neural environment lighting. In *Proceedings of the IEEE/CVF ICCV*. 79–89.
- Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. 2023. NeRO: Neural Geometry and BRDF Reconstruction of Reflective Objects from Multiview Images. *ACM Trans. Graph.* 42, 4, Article 114 (jul 2023), 22 pages.
- Morgan McGuire and Michael Mara. 2014. Efficient GPU screen-space ray tracing. *Journal of Computer Graphics Techniques (JCGT)* 3, 4 (2014), 73–85.
- Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of ECCV*. 405–421.
- Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4 (2022), 1–15.
- Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF CVPR*. 8280–8290.
- Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF CVPR*. 7495–7504.
- A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Niessner, J. T. Barron, G. Wetzstein, M. Zollhofer, and V. Golyanik. 2022. Advances in Neural Rendering. *Computer Graphics Forum* 41, 2 (2022), 703–735.
- Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: image synthesis using neural textures. *ACM Trans. Graph.* 38, 4, Article 66 (Jul 2019).
- Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *Proceedings of the IEEE/CVF CVPR*. IEEE, 5481–5490.
- Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum* 41, 2 (2022), 641–676.
- Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-NeRF: Point-based Neural Radiance Fields. *arXiv preprint arXiv:2201.08845* (2022).
- Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5752–5761.
- Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021a. Physgc: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF CVPR*. 5453–5462.
- Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. 2022a. Differentiable Point-Based Radiance Fields for Efficient View Synthesis. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea.). ACM, New York, NY, USA, Article 7.
- Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021b. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.* 40, 6 (2021).
- Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. 2022b. Modeling indirect illumination for inverse rendering. In *Proceedings of the IEEE/CVF CVPR*. 18643–18652.

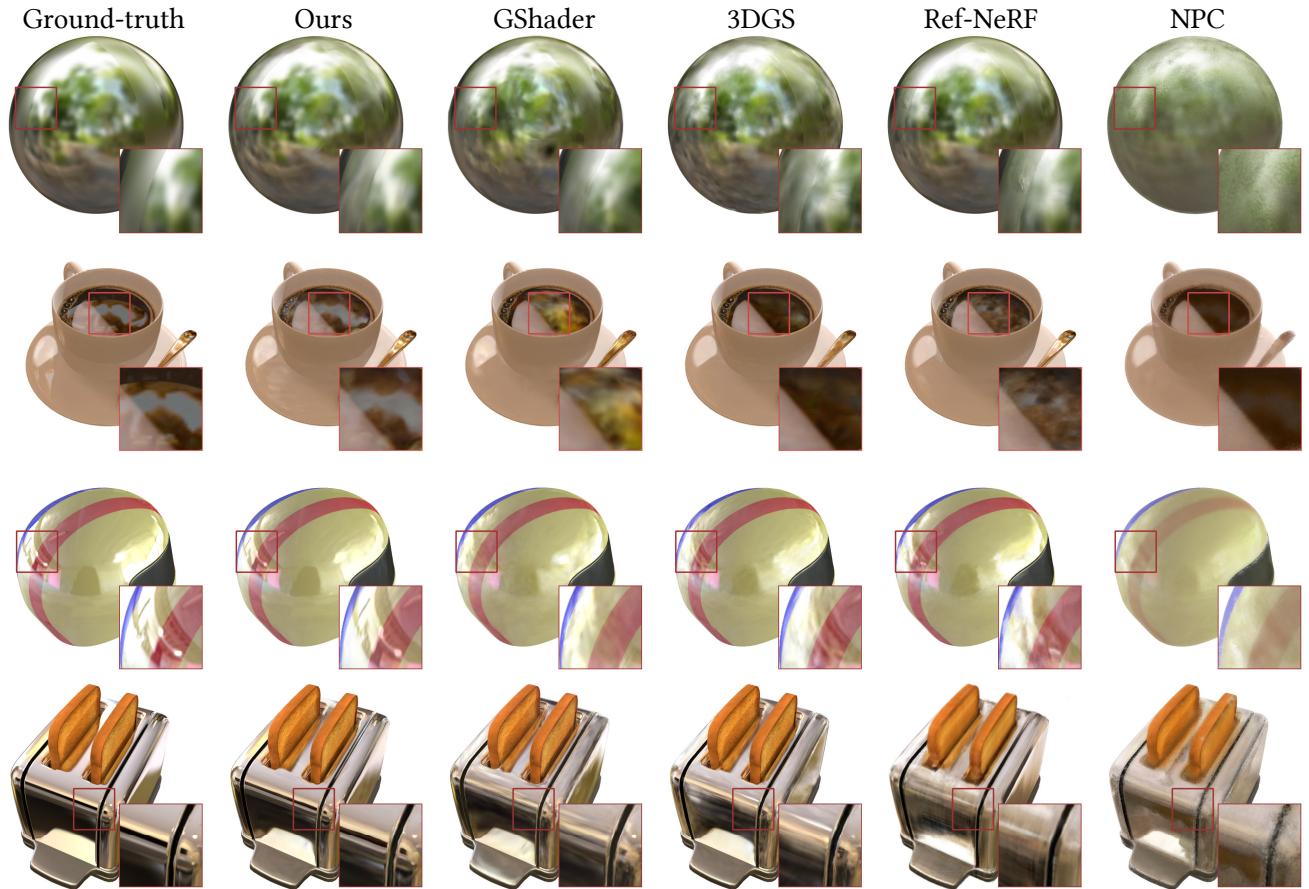


Figure 8: Qualitative comparisons on synthetic scenes. From top to bottom: ball, coffee, helmet, toaster.

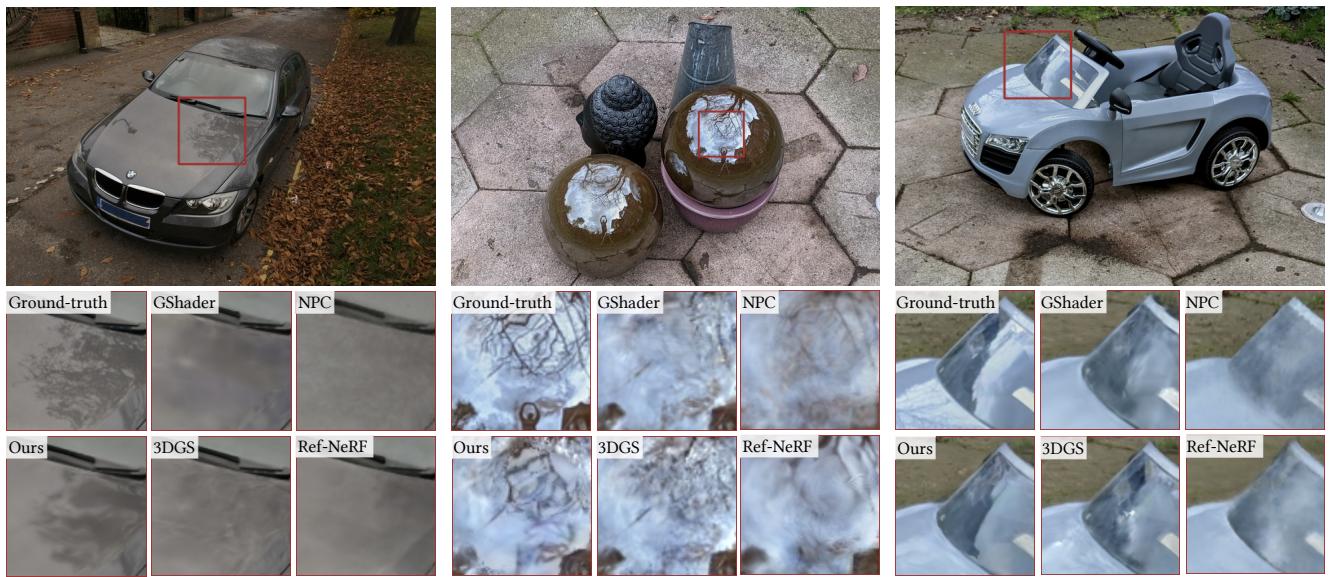
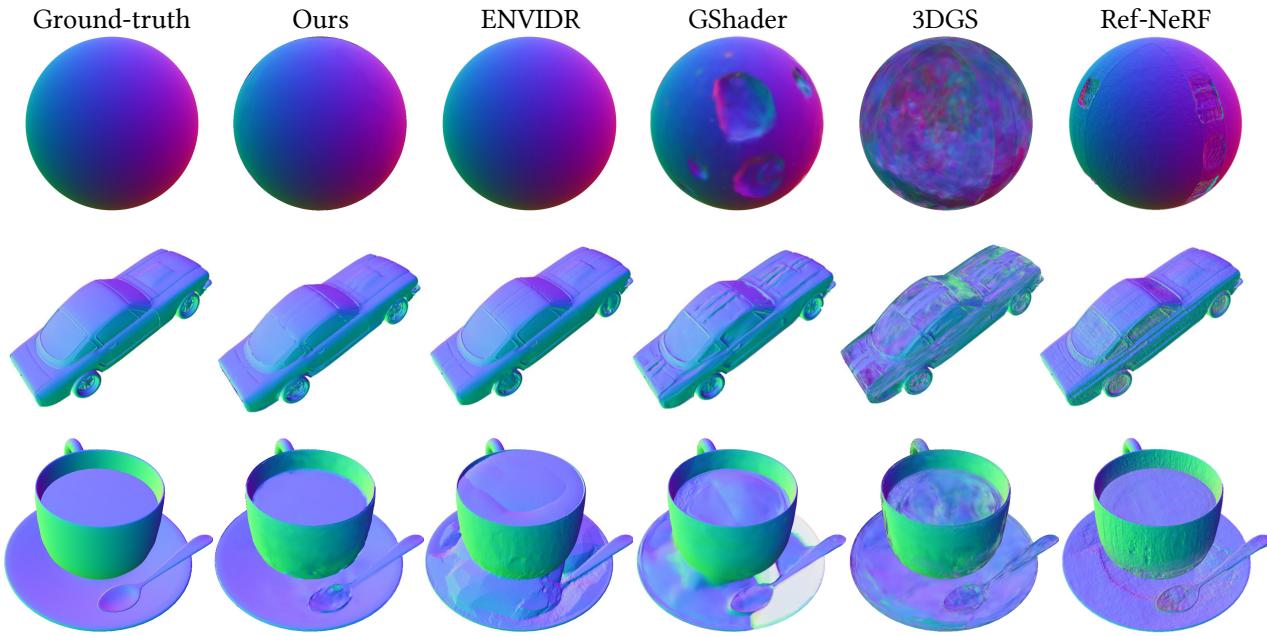


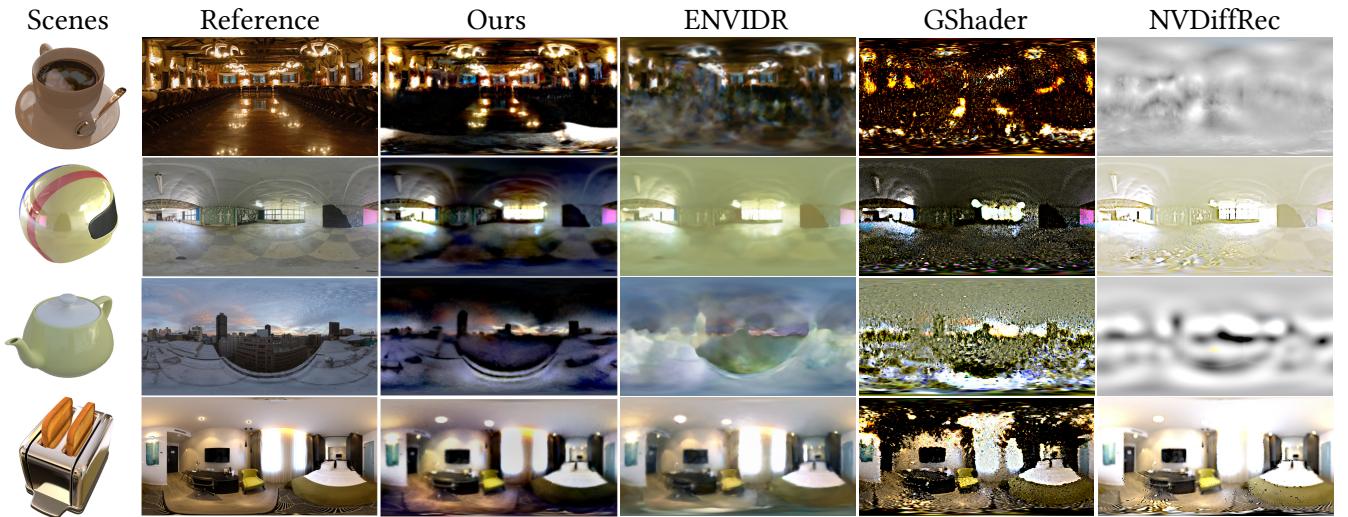
Figure 9: Qualitative comparisons on real scenes. From left to right: sedan, garden, toycar.



**Figure 10:** Qualitative comparisons between ENVIDR [Liang et al. 2023] and our method on luyu and coffee synthetic scenes.



**Figure 11:** Qualitative comparisons of normal produced by different methods.



**Figure 12:** Qualitative comparisons of environment maps estimated by different methods.