

EnvGS: Modeling View-Dependent Appearance with Environment Gaussian

Tao Xie^{1*} Xi Chen^{1*} Zhen Xu¹ Yiman Xie¹ Yudong Jin¹
 Yujun Shen² Sida Peng¹ Hujun Bao¹ Xiaowei Zhou^{1†}

¹ Zhejiang University ² Ant Group

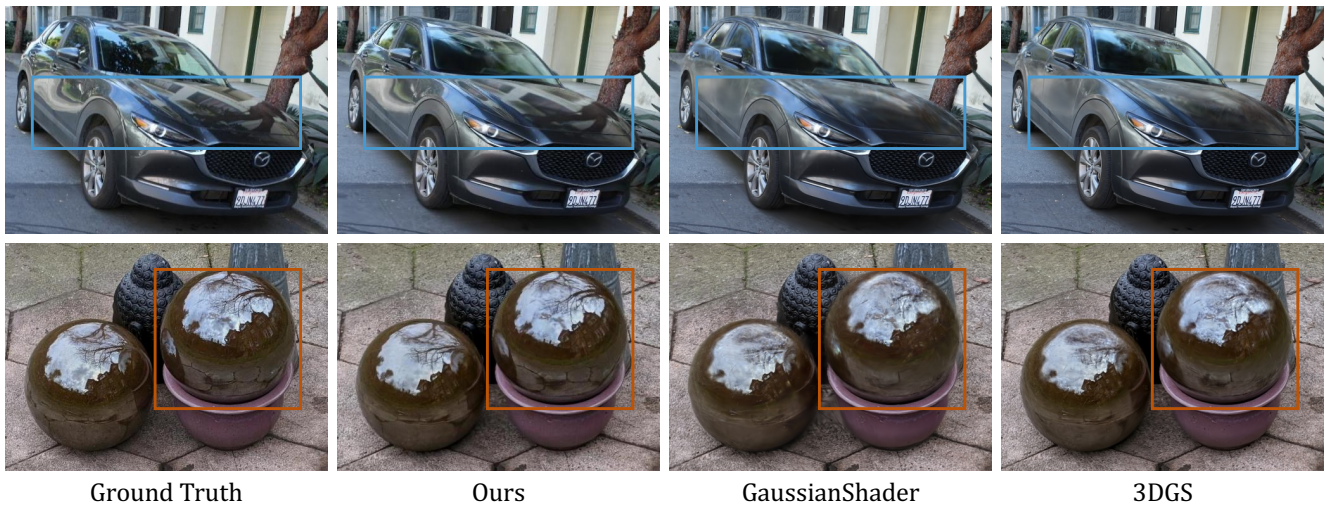


Figure 1. **Photorealistic, real-time rendering of real-world scenes with complex reflections.** Our proposed EnvGS outperforms prior works in capturing complex reflection effects, especially near-field reflections and high-frequency details while maintaining real-time rendering speed. Please see our supplementary video for better visualizations.

Abstract

Reconstructing complex reflections in real-world scenes from 2D images is essential for achieving photorealistic novel view synthesis. Existing methods that utilize environment maps to model reflections from distant lighting often struggle with high-frequency reflection details and fail to account for near-field reflections. In this work, we introduce EnvGS, a novel approach that employs a set of Gaussian primitives as an explicit 3D representation for capturing reflections of environments. These environment Gaussian primitives are incorporated with base Gaussian primitives to model the appearance of the whole scene. To efficiently render these environment Gaussian primitives, we developed a ray-tracing-based renderer that leverages the GPU’s RT core for fast rendering. This allows us to jointly optimize our model for high-quality reconstruction while maintain-

ing real-time rendering speeds. Results from multiple real-world and synthetic datasets demonstrate that our method produces significantly more detailed reflections, achieving the best rendering quality in real-time novel view synthesis. The code is available at <https://zju3dv.github.io/envgs>.

1. Introduction

Novel view synthesis aims to generate novel views of 3D scenes based on a set of input images, which enables many applications such as VR/AR, and autonomous driving. Recent advances in Neural Radiance Fields (NeRF) [28] have demonstrated impressive rendering performance. However, NeRF’s high computational cost makes it challenging for real-time applications. More recently, 3D Gaussian Splatting (3DGS) [17] explicitly models scenes with 3D Gaussian primitives and utilizes rasterization for rendering, achieving real-time rendering with competitive quality. However, modeling complex high-frequency specular

* Equal Contribution. † Corresponding author: Xiaowei Zhou

reflections remains challenging for 3DGS due to the limited expressiveness of the Spherical Harmonics (SH).

Recent works GaussianShader [15] and 3DGS-DR [51], enhance 3DGS by integrating an environment map and employing shading functions to blend the appearance from both the environment map and SH for the final rendering. While additional environmental lighting can enhance 3DGS’s reflection modeling ability, it still struggles to reconstruct complex specular reflections accurately due to two factors. First, the assumption of distant lighting in environment maps limits their ability to only capture distant illumination and difficult to synthesize accurate near-field reflections. Second, this representation inherently lacks sufficient capacity to capture high-frequency reflection details.

In this paper, we present EnvGS, a novel approach for modeling complex reflections in real-world scenes, addressing the aforementioned challenges. We propose to model reflections using a set of Gaussian primitives termed the *environment Gaussian*. The geometry and base appearance are represented by another set of Gaussian primitives called *base Gaussian*. We effectively blend the two Gaussians for rendering and optimization. Our rendering process begins with rendering the *base Gaussian* for the per-pixel surface position, normal, base color, and blending weight. Next, we render the *environment Gaussian* at the surface point in the direction of the reflection of the viewing direction around the surface normal to capture reflection colors. Finally, we blend the base color with the reflection color to achieve the final rendering results. In contrast to previous methods, EnvGS captures high-frequency reflection details using Gaussian primitives, offering superior modeling capabilities compared to environment maps. Additionally, our explicit 3D reflection representation eliminates the need for distant lighting assumptions, enabling accurate modeling of near-field reflections, as shown in Fig. 1.

To render the *environment Gaussian* at each intersection point along the reflection direction, we create a fully differentiable ray-tracing renderer for 2DGS since rasterization is not suited for this task. We build the ray-tracing renderer on CUDA and OptiX [31] for real-time rendering and efficient optimization of *environment Gaussian*. The rendering process starts by constructing a bounding volume hierarchy (BVH) from the 2D Gaussian primitives. We then cast rays against the BVH, gathering ordered intersections in chunks while integrating the Gaussian properties through volume rendering [16] to achieve the final results. Our Gaussian ray-tracing renderer enables detailed reflection rendering at real-time performance. Furthermore, it allows for efficient joint optimization of the *environment Gaussian* and *base Gaussian*, which is essential for accurate reflection modeling, as demonstrated in Sec 5.4.

To validate the effectiveness of our method, we evaluate EnvGS on several real and synthetic datasets. The re-

sults demonstrate that our methods achieve state-of-the-art performance in real-time novel view synthesis and considerably surpass existing real-time methods, particularly in synthesizing complex reflections in real-world scenes.

In summary, we make the following contributions:

- We propose a novel scene representation for accurately modeling complex near-field and high-frequency reflections in real-world environments.
- We developed a real-time ray-tracing renderer for 2DGS, enabling joint optimization of our representation for accurate scene reconstruction while achieving real-time rendering speeds.
- Extensive experiments shows that EnvGS significantly outperforms previous methods. To the best of our knowledge, EnvGS is the first method that achieves real-time photorealistic specular reflections synthesizing in real-world scenes.

2. Related Work

In computer vision and graphics research [9, 12, 20, 23, 40, 46, 50], our work falls into the area of learning scene representations from a set of posed RGB images. In this section, we review NeRF-based and Gaussian Splatting-based methods, particularly their handling of view-dependent effects.

Neural Radiance Field. NeRF [28] introduced the concept of neural radiance fields, which model scenes as implicit multilayer perceptrons (MLPs) and render them via volume rendering, achieving impressive results in novel view synthesis. Subsequent advancements focused on enhancing rendering quality [2–4] and computational efficiency. [6, 7, 24, 30, 35]. [21, 41, 42, 48] introduce Signed Distance Field into NeRF to improve geometry quality. However, these methods often model view-dependent effects via simple viewing directions, which can lead to blurry reflection renderings. To address this, Ref-NeRF [37] encodes the outgoing radiance using the reflected view direction, yielding improved results under distant lighting conditions. Follow-up works [8, 22, 25, 39] leverage Signed Distance Fields (SDF) to refine surface normals to enhance reflection and geometry quality. SpecNeRF [27] further incorporates spatially varying Gaussian directional encoding to better capture near-field reflections. NeRF-Casting [38] represents the scene in a unified manner, similar to [4], and performs ray marching along reflection directions to integrate reflection features, which are then decoded into color with MLPs. This approach achieves impressive results on real-world data under both near and distant lighting conditions. However, the requirement for multiple MLP queries per ray makes NeRF-Casting unsuitable for real-time rendering and necessitates substantial training time. In contrast, our approach delivers real-time rendering capabilities while significantly reducing training time.

Gaussian Splatting. Recently, 3D Gaussian Splatting (3DGS) [17] has made significant strides toward real-time novel view synthesis. Unlike volume-based rendering, Gaussian Splatting uses efficient rasterization to render spatial Gaussian kernels, achieving real-time performance at 1080P resolution. Mip-Splatting [52] further introduces a 3D smoothing filter and a 2D Mip filter for alias-free renderings. Scaffold-GS [26] introduces structured 3D Gaussians to improve rendering efficiency and quality. [5] introduces triple splitting for efficient relighting. 2D Gaussian Splatting (2DGS) [13] replaces the 3D Gaussian kernels with 2D Gaussian that better align with 3D surfaces, leading to more accurate surface reconstruction. However, these methods face challenges in accurately modeling reflections due to their use of spherical harmonics (SH) to parameterize view-dependent effects based solely on viewing direction, which often results in blurry reflections. More recent works GaussianShader [14] and 3DGS-DR [18] try to incorporate additional environment maps to improve reflection modeling ability. However, these methods only consider distant lighting, ignoring near-field lighting, and are unable to capture high-frequency reflection details. 3iGS [36] extends 3DGS by augmenting it with an illumination field using tensorial factorization, rendering final reflections through a neural renderer. However, it is limited to bounded scenes, which restricts its applicability in real-world, unbounded environments. Concurrent work [54] choose to model near-field reflection with tensorial factorization and far-field with spherical feature grid. Our method adopts 2DGS as the scene representation and models environmental illumination using an additional set of environment Gaussian, which is rendered via our proposed Gaussian tracer. Our approach inherently supports high-frequency, near-field, and distant lighting in unbounded scenes, enabling detailed reflection rendering while maintaining real-time performance.

3. Preliminary

We begin by introducing 2D Gaussian Splatting (2DGS) [13], which our approach is built upon. 2DGS is an explicit scene representation similar to 3D Gaussian Splatting (3DGS) [17], which uses Gaussian primitives and rasterization to render screen projections. The key difference is that 2DGS represents the scene with a set of scaled 2D Gaussian primitives defined in a local tangent plane within world space by a transformation matrix \mathbf{H} :

$$\mathbf{H} = \begin{bmatrix} s_u \mathbf{t}_u & s_v \mathbf{t}_v & 0 & \mathbf{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1)$$

where \mathbf{p}_k , $(\mathbf{t}_u, \mathbf{t}_v)$, and (s_u, s_v) denote the center, the two principal tangential vectors, and the scaling factors of the Gaussian, respectively.

To render an image, 2DGS employs the method described in [44] to determine the ray-primitive intersections.

These intersection points are subsequently utilized to compute the Gaussian’s contribution to the final image. The Gaussian properties are then integrated using a volume rendering algorithm to obtain the final pixel color:

$$\mathbf{c} = \sum_{i=1}^N T_i \alpha_i \mathbf{c}_i, \text{ with } \alpha_i = \sigma_i \mathcal{G}_i, T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (2)$$

where $\mathcal{G}(\cdot)$ is the standard 2D Gaussian value evaluation.

Compared to 3D Gaussian, 2D Gaussian offers distinct advantages as a surface representation. First, the ray-splat intersection method adopted by 2DGS avoids multi-view depth inconsistency. Second, 2D Gaussian inherently provides a well-defined normal, which is essential for high-quality surface reconstruction and accurate reflection calculations. However, 2DGS relies on the limited representational capacity of Spherical Harmonics (SH) to model scene appearance, preventing it from capturing strong view-dependent effects like specular reflections, which leads to poor rendering results and “foggy” geometry [37]. To this end, we use the geometry-aligned 2D Gaussian primitives as the base scene representation and demonstrate how we effectively model complex reflections in the next section.

4. Method

Given a set of input images of a reflective scene, our goal is to reconstruct the 3D scene and synthesize photorealistic novel views in real-time. To achieve this, we propose utilizing *environment Gaussian* as an explicit 3D environment representation, which enables accurate modeling of complex reflections in real-world scenes. Additionally, we represent the scene geometry and base colors using another set of Gaussian primitives, denoted as *base Gaussian*. In this section, we first detail how *environment Gaussian* and *base Gaussian* work together to model complex reflections within the scene (Sec. 4.1). Then, We describe the design of a ray-tracing renderer that leverages the GPU’s RT cores to efficiently render and optimize the *environment Gaussian* (Sec. 4.2). Finally, we discuss our optimization process. (Sec. 4.3). An overview of our method is in Fig. 2.

4.1. Reflective Scenes Modeling

Gaussian splatting [13, 17] models appearance using Spherical Harmonics (SH), which has limited representation capacity for view-dependent effects. These limitations hinder its ability to capture complex, high-frequency specular reflections. Building on this observation, our key insight is that modeling reflections with a Gaussian environment representation can better model complex reflection effects while significantly reducing the complexity required for each Gaussian to capture intricate details within its SH.

Our proposed reflective scene representation includes two sets of 2D Gaussians: a *base Gaussian* \mathbf{P}_{base} for mod-

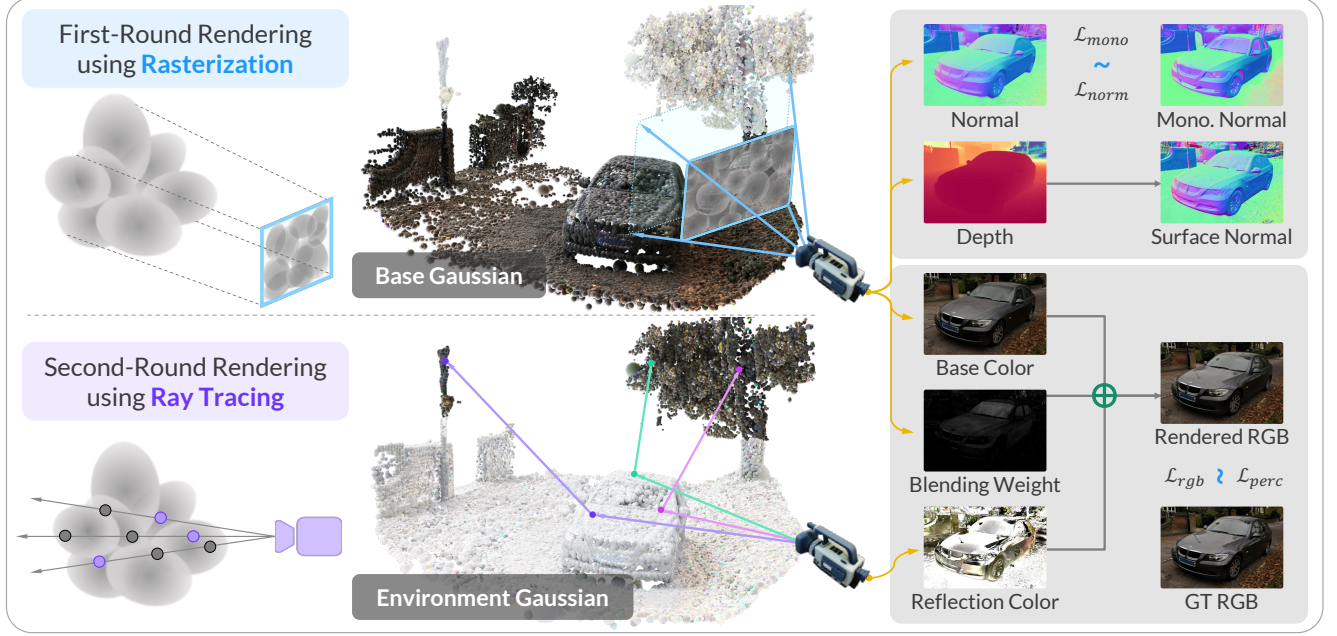


Figure 2. **Overview of EnvGS.** The rendering process begins by rasterizing the *base Gaussian* to obtain per-pixel normals, base colors, and blending weights. Next, we render the *environment Gaussian* in the reflection direction using our ray-tracing-based Gaussian renderer to capture the reflection colors. Finally, we combine the reflection and base colors for the final output. We jointly optimize the *environment Gaussian* and *base Gaussian* using monocular normals [49] and ground truth images for supervision.

eling the scene’s geometry and base appearance, and another *environment Gaussian* \mathbf{P}_{env} for capturing scene reflections. The basic parameterization of each Gaussian primitive is consistent with the original 2DGS [13], including 3D center position \mathbf{p} , opacity α , two principal tangential vectors $(\mathbf{t}_u, \mathbf{t}_v)$, a scaling vector (s_u, s_v) , and SH coefficients. To combine the two appearance components from the base and environment Gaussian into the final result, we introduce a blending weight for each base Gaussian.

The rendering process is performed in three steps. First, the base Gaussian \mathbf{P}_{base} is rendered using standard 2D Gaussian splatting to obtain the base color \mathbf{c}_{base} . By applying the volume rendering integration using Eq. (2), we also derive the surface position \mathbf{x} , surface normals \mathbf{n} , and blending weight β as:

$$v = \sum_{i \in \mathcal{N}} v_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), v \in \{\mathbf{x}, \mathbf{n}, \beta\}. \quad (3)$$

Then, we compute the reflection direction \mathbf{d}_{ref} based on the camera ray direction \mathbf{d}_{cam} and the surface normal \mathbf{n} :

$$\mathbf{d}_{ref} = \mathbf{d}_{cam} - 2(\mathbf{d}_{cam} \cdot \mathbf{n})\mathbf{n}. \quad (4)$$

With the reflection direction \mathbf{d}_{ref} and surface point \mathbf{x} , the environment Gaussian \mathbf{P}_{env} is rendered using our differentiable Gaussian tracer to obtain the reflection color \mathbf{c}_{ref} , as detailed in Sec. 4.2. The final color is obtained through:

$$\mathbf{c} = (1 - \beta) \cdot \mathbf{c}_{base} + \beta \cdot \mathbf{c}_{ref}. \quad (5)$$

We blend the base color \mathbf{c}_{base} and reflection color \mathbf{c}_{ref} using the blending weight β . A visualization of \mathbf{c}_{base} and \mathbf{c}_{ref} can be found at Fig. 3.

Discussion. Compared to the environment map used by Gaussianshader [15] and 3DGS-DR [51], our explicit Gaussian environment representation offers several advantages. First, our method more accurately captures near-field reflections caused by occlusions from nearby objects. This improvement arises from explicitly modeling each Gaussian at its exact spatial location, thus avoiding the ambiguities and inaccuracies inherent in environment map representations that assume distant lighting. Second, by utilizing Gaussian primitives, EnvGS’s environment representation achieves greater expressiveness than low-frequency environment maps, enabling the capture of finer reflection details and enhancing rendering quality, as demonstrated by our experiments 5.3.

4.2. Differentiable Ray Tracing

Rendering the environment Gaussian with rasterization is impractical, as each pixel corresponds to a unique reflection ray and functions as a virtual camera. To address this, we draw on [29] and leverage the advanced optimizations of modern GPUs to design a novel, fully differentiable ray tracing framework. Built on OptiX [31], our framework achieves real-time rendering of 2,000,000 2DGS with a resolution of 1292x839 at 30 FPS on an RTX 4090 GPU.

In order to fully utilize the hardware acceleration for



Figure 3. **Visualization of reflection and base color.** Our method successfully reconstructs near-field and distant reflections using the *environment Gaussian* instead of baking into the base color.

ray-primitive intersections, we need to convert each 2D Gaussian into a geometric primitive compatible with GPU processing and insert it into a bounding volume hierarchy (BVH). In light of this, we propose to represent each 2D Gaussian with two triangles. Specifically, we first define the four Gaussian bounding vertices $\mathbf{V}_{local} = \{(\text{sgn}(r), \text{sgn}(r))\}$ in the local tangent plane, where $\text{sgn}(\cdot)$ is the sign function and r is set to 3 representing three times the sigma range. Then, the four local bounding vertices \mathbf{V}_{local} are transformed to world space as the vertices of the two triangles covering the Gaussian \mathbf{V}_{world} using Eq. 1. After the transformation, the triangles are organized into a BVH, which serves as the input for the ray tracing process.

We develop a custom CUDA kernel using the *raygen* and *anyhit* programmable entry points of OptiX. Inspired by [29], the rendering is done in a chunk-by-chunk manner. The *anyhit* kernel traces the input ray to obtain a chunk of size k , while *raygen* integrates this chunk and invokes *anyhit* to retrieve the next chunk along the ray. Specifically, consider an input ray with origin \mathbf{o} and direction \mathbf{d} . The *raygen* program first initiates a traversal against the BVH to identify all possible intersections along the ray. During the traversal, the *anyhit* program sorts each intersected Gaussian by the depth and maintains a sorted k -buffer for the closest k intersections. We empirically found that k with 16 is the best trade-off between traversal counts and the number of Gaussians sorted per traversal. After traversal, the *raygen* program integrates properties of the sorted Gaussians in the buffer following Eq. 2. The Gaussian response is calculated by applying the inverse of the transformation matrix \mathbf{H} to the ray’s intersection point \mathbf{x}_i and evaluating the Gaussian value at the transformed point as:

$$\mathcal{G}_i(\mathbf{u}_i) = \mathcal{G}_i(\mathbf{H}^{-1}\mathbf{x}_i). \quad (6)$$

This process repeats until no further intersections are found

along the ray or the accumulated transmittance drops below a specified threshold. More details can be found in supplementary materials.

Our ray tracing framework is fully differentiable, allowing for end-to-end optimization of both the base and the environment Gaussian primitives. However, storing all intersections during the forward pass and performing the backward pass in back-to-front order, as done in 3DGS [17], is impractical due to high memory consumption. To address this, we implement the backward pass in the same front-to-back order as the forward pass by re-casting rays and computing gradients for each integration step. A key aspect is calculating the gradient with respect to the input ray origin $\frac{\partial \mathcal{L}}{\partial \mathbf{o}}$ and direction $\frac{\partial \mathcal{L}}{\partial \mathbf{d}}$, which is crucial for the joint optimization of our model (see supplementary for more details). Our ablation in Sec 5.4 demonstrates that this joint optimization of geometry and appearance is essential for recovering geometrically accurate surfaces.

4.3. Optimization

To enhance training stability, we initiate optimization by first training the base Gaussian \mathbf{P}_{base} , which is initialized using the sparse point cloud obtained from Structure-from-Motion (SfM) [11, 34]. After the bootstrapping phase, we initialize the environment Gaussian \mathbf{P}_{env} by partitioning the scene’s bounding box \mathbf{B}_{scene} into N^3 sub-grids and randomly sampling K primitives within each grid, and then optimize the base Gaussian and environment Gaussian jointly. The scene’s bounding box \mathbf{B}_{scene} is determined by the 99.5% quantile of the sparse point cloud’s bounding box \mathbf{B}_{sfm} obtained from SfM, the sub-grids resolution N is set to 32, and K is set to 5 for each grid.

We follow 2DGS [13] to add a normal consistency constraint between the rendered normal map \mathbf{n} and the gradients of the depth map \mathbf{N}_d :

$$\mathcal{L}_{norm} = \frac{1}{N_p} \sum_{i=1}^{N_p} (1 - \mathbf{n}_i^\top \mathbf{N}_d), \quad (7)$$

where N_p is the number of pixels in the image and \mathbf{N}_d is calculated as finite differences of neighboring pixels in the depth map:

$$\mathbf{N}_d(\mathbf{u}) = \frac{\nabla_u \mathbf{p}_d \times \nabla_v \mathbf{p}_d}{\|\nabla_u \mathbf{p}_d \times \nabla_v \mathbf{p}_d\|}. \quad (8)$$

However, the normal consistency constraint alone is insufficient for accurately modeling ambiguous surfaces involving both reflection and refraction. Inspired by recent advances in monocular normal estimation [1, 10, 49], we propose to supervise the rendered normal map \mathbf{n} using monocular normal estimates \mathbf{N}_m :

$$\mathcal{L}_{mono} = \frac{1}{N_p} \sum_{i=1}^{N_p} (1 - \mathbf{n}_i^\top \mathbf{N}_m). \quad (9)$$

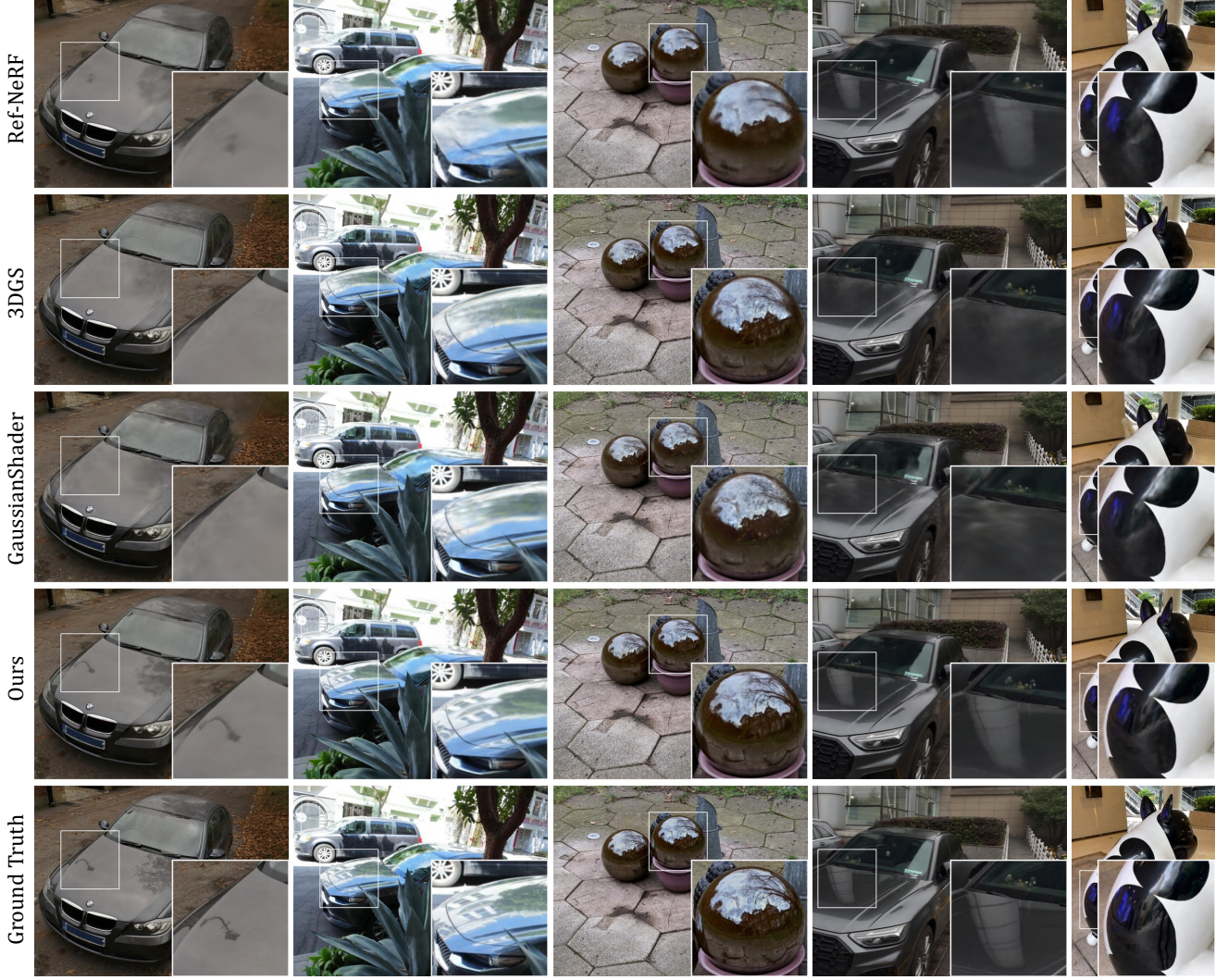


Figure 4. **Qualitative comparison on real scenes.** Our method significantly improves rendering quality over previous approaches, particularly in producing more detailed reflections. Zoom in for more details.

In addition to the \mathcal{L}_1 image loss and D-SSIM loss \mathcal{L}_{ssim} employed by 3DGS [17], we also use perceptual loss [53] to enhance the perceived quality of the rendered image:

$$\mathcal{L}_{perc} = \|\Phi(\mathbf{I}) - \Phi(\mathbf{I}_{gt})\|_1, \quad (10)$$

where Φ is the pre-trained VGG-16 network [33] and $\mathbf{I}, \mathbf{I}_{gt}$ are the rendered and ground truth images, respectively.

The final loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_1 \mathcal{L}_{norm} + \lambda_2 \mathcal{L}_{mono} + \lambda_3 \mathcal{L}_{perc}, \quad (11)$$

where \mathcal{L}_{rgb} is the photometric reconstruction loss combining the \mathcal{L}_1 loss and a D-SSIM [43] term with a ratio of 0.8 and 0.2 respectively. We set $\lambda_1 = 0.04$, $\lambda_2 = 0.01$, and $\lambda_3 = 0.01$ across our experiments.

5. Experiments

5.1. Implementation Details

We implement EnvGS with custom OptiX kernels and optimize our model using the PyTorch framework [32, 47] with the Adam optimizer [19]. Specifically, we set the learning rates for the parameters of each base and environment Gaussian to match those used in 2DGS [13]. The learning rate for the blending weight is set to $1e^{-2}$. During training, we apply the adaptive Gaussian control strategy of 3DGS [17] with the normal propagation and color sabotage introduced in 3DGS-DR [18]. Since our Gaussian tracer integrates the Gaussian properties directly in 3D space, there is no valid gradient for the projected 2D center, which is used as the densification criterion in 3DGS. Following [29], we accumulate the 3D spatial gradients of the Gaussian position to achieve a similar effect. Note that each accumulated gra-

Methods		Ref-NeRF Real Scenes [37]			NeRF-Casting Scenes [38]			FPS ↑	Training Time ↓
		PSNR↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓		
Non real-time	Ref-NeRF* [37]	23.087	0.625	0.261	30.583	0.890	0.124	<0.1	47h
	UniSDF [39]	23.700	0.635	0.266	30.838	0.889	0.130	<0.1	>47h
	ZipNeRF [4]	23.677	0.635	0.247	31.740	0.904	0.105	<0.1	>47h
	NeRF-Casting [38]	24.670	0.659	0.246	31.023	0.889	0.128	<0.1	>47h
Real-time	3DGS [17]	23.700	0.641	0.262	28.860	0.877	0.159	182.307	0.6h
	2DGS [13]	23.804	0.654	0.281	28.276	0.862	0.193	159.188	0.7h
	GaussianShader [14]	22.875	0.622	0.314	26.412	0.835	0.216	27.945	1.6h
	3DGS-DR [51]	23.522	0.640	0.274	28.487	0.858	0.197	133.593	1.0h
	Ours	24.617	0.671	0.241	30.444	0.886	0.148	26.221	2.5h

Table 1. **Quantitative comparison on Ref-Real [37] and NeRF-Casting Scenes [38] datasets.** Our method delivers the highest rendering quality among real-time techniques and outperforms several non-real-time methods, achieving competitive results with the state-of-the-art non-real-time method NeRF-Casting [38], while being 100 times faster. Note that Ref-NeRF* is an improved version of Ref-NeRF [37] that uses Zip-NeRF’s geometry model. All metrics are evaluated at 1/4 resolution as prior works [38].

Methods	Foreground			Near-Field		
	PSNR↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [17]	31.681	0.955	0.046	44.161	0.994	0.008
2DGS [13]	31.328	0.953	0.053	43.624	0.994	0.010
GaussianShader [14]	30.694	0.947	0.058	42.391	0.993	0.010
3DGS-DR	31.814	0.953	0.050	44.007	0.994	0.009
Ours	33.295	0.962	0.040	46.392	0.996	0.007

Table 2. **Quantitative results of foreground and near-field region on Ref-Real [37] and NeRF-Casting Scenes [38].** See supplementary Appendix A.1 for more details and qualitative results.

dient is scaled by half of the intersection depth to prevent under-densification in distant regions. All experiments are conducted on a single NVIDIA RTX 4090 GPU.

5.2. Datasets and Evaluation Metrics

We train and evaluate EnvGS on a range of datasets with a focus on real-world scenes characterized by complex view-dependent effects. We evaluate the Ref-Real [37] and NeRF-Casting Shiny Scenes [38] to demonstrate our method’s ability for complex specular reflections in real-world scenes. We additionally captured two more real-world scenes for a more comprehensive evaluation. We demonstrate that our methods can reconstruct detailed reflections on complex real-world scenes with real-time rendering speed. Additionally, we evaluate our method on the synthetic Shiny Blender dataset [37], which is rendered using environment maps. More evaluation results can be found in the supplementary materials.

We maintain consistent training and testing splits and image resolution across all datasets, following prior works [3, 17, 37]. We use three commonly used metrics for evaluation: PSNR, SSIM [43], and LPIPS [53].

5.3. Baseline Comparisons

We compare our method with both implicit and explicit prior works, including Zip-NeRF [13], 3DGS [17], 2DGS [13], which are designed for general scenes with pri-

Methods	Audi			Dog		
	PSNR↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Ref-NeRF [37]	24.529	0.713	0.397	23.620	0.673	0.389
3DGS [17]	26.171	0.825	0.181	25.300	0.882	0.165
2DGS [13]	25.869	0.819	0.198	24.933	0.876	0.184
GaussianShader [14]	24.768	0.795	0.225	23.061	0.840	0.226
3DGS-DR [51]	26.093	0.820	0.192	23.943	0.850	0.210
Ours	28.629	0.876	0.121	25.599	0.885	0.151

Table 3. **Quantitative results on our self-captured scenes.**

marily diffuse appearances, as well as with methods specifically tailored for scenes with strong specular reflections, including Ref-NeRF [37], GaussianShader [14], 3DGS-DR [18], and NeRF-Casting [38]. We also compare with ENVIDR [22], NDE [45] these two object-level methods.

In this section, we present quantitative and qualitative results to illustrate the advantages of our methods. Since accurate reflection reconstruction and rendering are the essential advantages of our method, we encourage readers to refer to the various rendered continuous video results in the supplementary material for a more comprehensive evaluation of our method.

We first evaluate our method on nine real-world scenes from the Ref-Real dataset [37], NeRF-Casting Shiny Scenes [38] and our self-captured real-world scenes, which feature complex geometry and specular reflections. The quantitative results, shown in Tab. 1 and Tab. 3, demonstrate that our method outperforms all explicit methods by a large margin and achieves comparable results to the current state-of-the-art implicit method, NeRF-Casting, while being significantly faster. The qualitative results in Fig. 4 further highlight the superior performance of our method in capturing complex view-dependent effects, especially near-field reflections and high-frequency details.

We provide additional comparisons on the Mip-NeRF 360 dataset [3], the Shiny Blender dataset [37] and with ENVIDR [22], NDE [45] in supplementary Appendix A, which also includes per-scene metric breakdowns from Tab. 1.

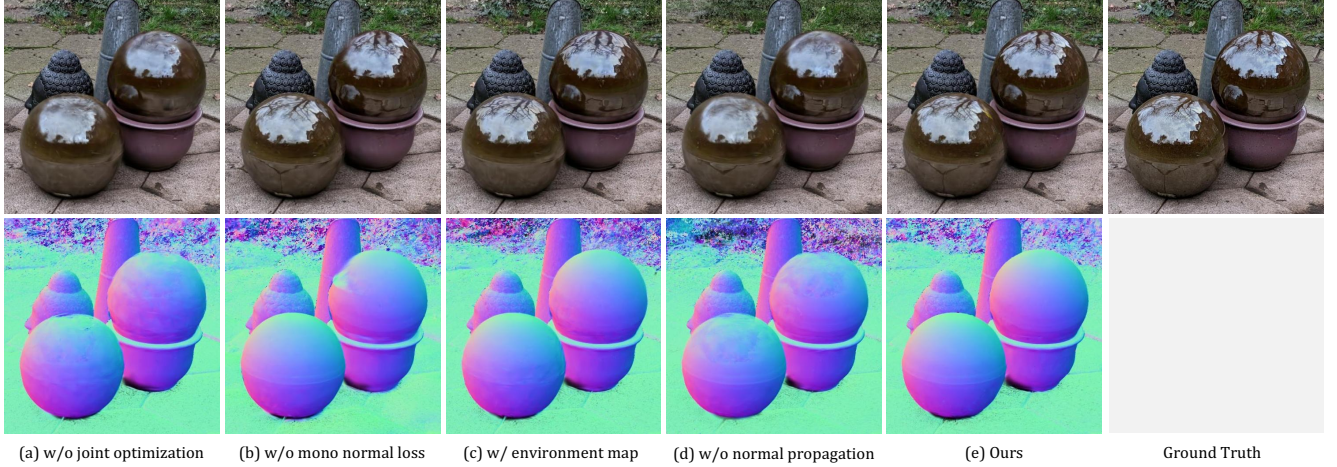


Figure 5. **Ablation study of proposed components on the Ref-Real dataset [37].** Removing either the monocular normal constraint or the joint optimization of base and environment Gaussians results in noisy geometry and inaccurate reflection reconstruction. The “w/ environment map” variant fails to capture near-field reflections.

Our method achieves competitive results on both datasets, demonstrating its versatility and effectiveness in handling diverse scenes with complex view-dependent effects.

5.4. Ablation Studies

	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
w/o joint optimization	24.034	0.644	0.287
w/o mono normal loss	24.107	0.648	0.270
w/ environment map	24.145	0.646	0.285
w/o color sabotage	24.268	0.650	0.269
w/o normal propagation	24.192	0.652	0.271
w/o lpips loss	24.567	0.671	0.280
Ours	24.617	0.671	0.241

Table 4. **Ablation studies.**

In this section, we conduct ablation studies of our key components on the Ref-Real dataset. Quantitative and qualitative results are shown in Tab. 4 and Fig. 5, respectively.

Joint optimization. The “w/o joint optimization” variant detaches the joint optimization of the *base Gaussian* and *environment Gaussian* from the reflection rendering step. As shown in Tab. 4 and Fig. 5, This variant fails to recover accurate geometry, leading to inferior reflection reconstruction and rendering quality.

Monocular normal constraint The “w/o monocular normal” variant removes the monocular normal constraint as described in Sec. 4.3. Training may become trapped in largely incorrect geometry, resulting in inaccurate reflection reconstruction, as demonstrated in Tab.4 and Fig.5.

Environment map representation The “w/ environment map” variant replaces our core Gaussian environment representation with an environment map representation while keeping all other components unchanged. Figure 5 illustrates that while it effectively captures smooth distant reflections, it has difficulty modeling near-field and high-frequency reflections, and produces more bumpy geometry.

Color sabotage and normal propagation. The “w/o color sabotage” and “w/o normal propagation” variants omit the color sabotage and normal propagation steps from our method, respectively. As demonstrated in Tab. 4, both variants result in reduced rendering quality.

Perceptual loss. The “w/o lpips loss” variants removes the perceptual loss. The results in Tab. 4 demonstrate that our method still produces detailed and accurate reflections without perceptual loss, which offers only marginal improvements.

6. Conclusion and Discussion

This paper introduced EnvGS, a novel reflective scene representation for high-quality complex reflection capturing and real-time rendering. Our method explicitly models reflections with a set of *environment Gaussian* primitives. The environment Gaussian primitives are used together with a set of base Gaussian primitives that model basic scene properties (geometry, base color, and blending weight) to model the appearance of the whole scene. Furthermore, we develop a differentiable Gaussian ray tracer utilizing GPU’s RT core to effectively optimize and render the *environment Gaussian*. The proposed method demonstrates superior performance in capturing complex reflections across various datasets.

A limitation of EnvGS is its difficulty with transparent and refractive materials, as it only addresses reflection direction. Future work could explore extending our method to accommodate these materials.

Acknowledgements. This work was partially supported by NSFC (No. U24B20154, 62402427), Ant Group, and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

- [1] Gwangbin Bae and Andrew J. Davison. Rethinking inductive biases for surface normal estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 5
- [2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 2
- [3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 7, 1, 8
- [4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 2, 7, 1, 6, 8
- [5] Zoubin Bi, Yixin Zeng, Chong Zeng, Fan Pei, Xiang Feng, Kun Zhou, and Hongzhi Wu. Gs3: Efficient relighting with triple gaussian splatting. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–12, 2024. 3
- [6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 2
- [7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2
- [8] Chen Guangcheng, He Yicheng, He Li, and Zhang Hong. Pidr: Polarimetric neural implicit surface reconstruction for textureless and specular objects. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 2
- [9] Yang Hang, Chen Rui, An Shipeng, Wei Hao, and Zhang Heng. The growth of image-related three dimensional reconstruction techniques in deep learning-driven era: a critical summary, 2023. 2
- [10] Jing He, Haodong Li, Wei Yin, Yixun Liang, Leheng Li, Kaiqiang Zhou, Hongbo Liu, Bingbing Liu, and Ying-Cong Chen. Lotus: Diffusion-based visual foundation model for high-quality dense prediction. *arXiv preprint arXiv:2409.18124*, 2024. 5
- [11] Xingyi He, Jiaming Sun, Yifan Wang, Sida Peng, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Detector-free structure from motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21594–21603, 2024. 5
- [12] Fuyuan Hu, Chenlu Li, Tao Zhou, Hongfu Cheng, and Minming Gu. A survey on point cloud completion algorithms for deep learning, 2025. 2
- [13] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3, 4, 5, 6, 7, 8
- [14] 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. *arXiv preprint arXiv:2311.17977*, 2023. 3, 7, 6, 8
- [15] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussian-shader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024. 2, 4, 1
- [16] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. 2
- [17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 3, 5, 6, 7, 2, 8
- [18] Ye Keyang, Hou Qiming, and Zhou Kun. 3d gaussian splatting with deferred reflection. 2024. 3, 6, 7, 1
- [19] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6
- [20] Yingqun Li, Xiao Hu, Xiang Xu, Yanning Xu, and Lu Wang. Deep learning-based foveated rendering in 3d space: a review, 2024. 2
- [21] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [22] Ruofan Liang, Huiting Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. Envidr: Implicit differentiable renderer with neural environment lighting. *arXiv preprint arXiv:2303.13022*, 2023. 2, 7
- [23] Haotong Lin, Sida Peng, Jingxiao Chen, Songyou Peng, Jiaming Sun, Minghuan Liu, Hujun Bao, Jiashi Feng, Xiaowei Zhou, and Bingyi Kang. Prompting depth anything for 4k resolution accurate metric depth estimation. 2025. 2
- [24] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 2
- [25] 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. In *SIGGRAPH*, 2023. 2
- [26] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 3
- [27] Li Ma, Vasu Agrawal, Haithem Turki, Changil Kim, Chen Gao, Pedro Sander, Michael Zollhöfer, and Christian Richardt. Specnerf: Gaussian directional encoding for specular reflections, 2023. 2
- [28] 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 *ECCV*, 2020. 1, 2
- [29] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Ricardo 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 and SIGGRAPH Asia*, 2024. 4, 5, 6
- [30] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multires-

- olution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. [2](#)
- [31] Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, et al. Optix: a general purpose ray tracing engine. *Acm transactions on graphics (tog)*, 29(4):1–13, 2010. [2](#), [4](#)
- [32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. [6](#)
- [33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [6](#)
- [34] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. [5](#)
- [35] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. [2](#)
- [36] Zhe Jun Tang and Tat-Jen Cham. 3igs: Factorised tensorial illumination for 3d gaussian splatting. *arXiv preprint arXiv:2408.03753*, 2024. [3](#), [7](#)
- [37] 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. *CVPR*, 2022. [2](#), [3](#), [7](#), [8](#), [1](#), [6](#)
- [38] 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. *arXiv preprint arXiv:2405.14871*, 2024. [2](#), [7](#), [1](#), [6](#), [8](#)
- [39] Fangjinhua Wang, Marie-Julie Rakotosaona, Michael Niemeyer, Richard Szeliski, Marc Pollefeys, and Federico Tombari. Unisdf: Unifying neural representations for high-fidelity 3d reconstruction of complex scenes with reflections. In *NeurIPS*, 2024. [2](#), [7](#), [6](#), [8](#)
- [40] Jinke Wang, Xingxing Zuo, Xiangrui Zhao, Jiajun Lyu, and Yong Liu. Review of multi-source fusion slam: current status and challenges, 2022. [2](#)
- [41] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. [2](#)
- [42] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. [2](#)
- [43] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. [6](#), [7](#)
- [44] Tim Weyrich, Simon Heinzle, Timo Aila, Daniel B Fasnacht, Stephan Oetiker, Mario Botsch, Cyril Flaig, Simon Mall, Kaspar Rohrer, Norbert Felber, et al. A hardware architecture for surface splatting. *ACM Transactions on Graphics (TOG)*, 26(3):90–es, 2007. [3](#)
- [45] Liwen Wu, Sai Bi, Zexiang Xu, Fujun Luan, Kai Zhang, Iliyan Georgiev, Kalyan Sunkavalli, and Ravi Ramamoorthi. Neural directional encoding for efficient and accurate view-dependent appearance modeling. In *CVPR*, 2024. [7](#), [2](#)
- [46] Jiankai Xing and Kun Xu. Physically based differentiable rendering: a survey, 2024. [2](#)
- [47] Zhen Xu, Tao Xie, Sida Peng, Haotong Lin, Qing Shuai, Zhiyuan Yu, Guangzhao He, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. Easyvolcap: Accelerating neural volumetric video research. In *SIGGRAPH Asia 2023 Technical Communications*, pages 1–4. 2023. [6](#)
- [48] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. [2](#)
- [49] 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. *ACM Transactions on Graphics*, 2024. [4](#), [5](#)
- [50] Hanqiao Ye, Yangdong Liu, and Shuhan Shen. Lightweight visual-based localization technology, 2024. [2](#)
- [51] Keyang Ye, Qiming Hou, and Kun Zhou. 3d gaussian splatting with deferred reflection. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–10, 2024. [2](#), [4](#), [7](#), [1](#), [6](#), [8](#)
- [52] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. [3](#)
- [53] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. [6](#), [7](#)
- [54] Youjia Zhang, Anpei Chen, Yumin Wan, Zikai Song, Junqing Yu, Yawei Luo, and Wei Yang. Ref-gs: Directional factorization for 2d gaussian splatting. *arXiv preprint arXiv:2412.00905*, 2024. [3](#)