3D Gaussian Inverse Rendering with Approximated Global Illumination

Zirui Wu[1,2] **Jianteng Chen**[2] **Laijian Li**[2] **Shaoteng Wu**[2] **Zhikai Zhu**[2]
Kang Xu[2] **Martin R. Oswald**[3] **Jie Song**[1,4]
[1] HKUST(GZ) [2] NIO [3] University of Amsterdam [4] HKUST

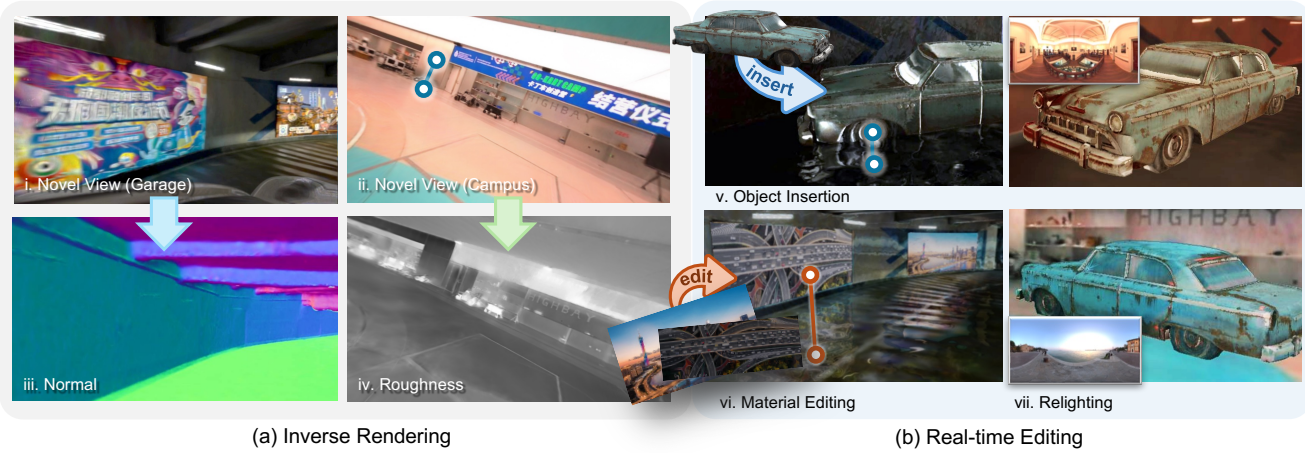

Figure 1. **Overview**: (a) Our inverse rendering pipeline recovers geometry and material properties from 3D captures, visualized through normal (iii) and roughness (iv) maps. The decomposition enables various editing capabilities: object insertion, material editing, and relighting. Our screen-space ray tracing technique ensures physically plausible reflections (visualized through corresponding point pairs).

Abstract

3D Gaussian Splatting shows great potential in reconstructing photo-realistic 3D scenes. However, these methods typically bake illumination into their representations, limiting their use for physically-based rendering and scene editing. Although recent inverse rendering approaches aim to decompose scenes into material and lighting components, they often rely on simplifying assumptions that fail when editing. We present a novel approach that enables efficient global illumination for 3D Gaussians Splatting through screen-space ray tracing. Our key insight is that a substantial amount of indirect light can be traced back to surfaces visible within the current view frustum. Leveraging this observation, we augment the direct shading computed by 3D Gaussians with Monte-Carlo screen-space ray-tracing to capture one-bounce indirect illumination. In this way, our method enables realistic global illumination without sacrificing the computational efficiency and editability benefits of 3D Gaussians. Through experiments, we show that the screen-space approximation we utilize allows for indirect illumination and supports real-time rendering and editing. Project page: https://wuzirui.github.io/gs-ssr.

1. Introduction

Creating digital replicas of the physical world that support realistic simulations is a fundamental challenge in 3D computer vision and graphics. Physically-based rendering engines like Blender [12] enable the creation of virtual environments that adhere to real-world physics, allowing us to edit and simulate scenarios as if they truly existed in the physical world. For instance, autonomous driving researchers may need to insert cars into reconstructed scenes or modify lighting conditions (as in Fig. 1-b) to test their perception algorithms. With recent advances in neural rendering, especially Neural Radiance Fields [3, 31, 33, 60] and 3D Gaussian Splattings [5, 18, 47, 50, 51], we can now reconstruct the 3D world and render photorealistic images using the reconstructed model.

Among these advances, Gaussian Splatting is particularly promising for simulation environments, thanks to its discrete nature and efficient rendering capabilities. Representing scenes as individual Gaussian primitives naturally facilitates local editing operations, and its rasterization-based pipeline enables real-time performance. However, current methods primarily focus on reconstruction, where illumination and material properties are **baked** into their

representation. This results in poor view extrapolation and, crucially for simulation purposes, the inability to edit material properties, scene geometry, and lighting conditions while maintaining physically plausible results (Fig. 1-b).

Recent inverse rendering techniques attempt to address these limitations by recovering explicit material and lighting properties [2, 26]. These methods combine physically based rendering theory with Gaussian Splats [19, 44, 56]. However, most existing works [19, 44] only consider direct illumination - light arriving directly from sources like the sun or lamps. They typically approximate this using environment maps that assume all light originates from an infinite distance. This simplification breaks down in real-world scenes where indirect illumination - light bouncing between surfaces before reaching our eyes - plays a crucial role. As shown in Fig. 1-(ii), these indirect effects are ubiquitous, from reflections of billboards on the ground to subtle color bleeding between walls. Without modeling these complex light interactions, current approaches struggle to achieve realistic rendering results in such environments.

Incorporating global illumination effects into Gaussian Splatting is desirable but presents certain technical considerations. While Gaussian Splatting produces photorealistic results efficiently [5], its rasterization-based approach differs from ray tracing methods - instead of actively exploring points in 3D space, it passively receives points that are projected onto the canvas, making it challenging to track reflection paths and compute indirect illumination.

Previous works have attempted to address these challenges through various simplifying assumptions, but this often comes at the cost of real-time editability. For instance, GaussianShader [19] operates under an object-centric model, using a global environment map to represent all incident lights. While some studies [14, 16, 56] tackle global illumination by pre-computing light-surface interactions into specific data structures, such as volumetric grids of indirect lights, these precomputed resources become invalid after any scene edits. This necessitates costly recomputation and ultimately limits real-time performance.

In this work, we bypass costly pre-computations and recomputations by utilizing rasterized G-buffers that store per-pixel geometric and material properties (Fig. 1 iii,iv). Our key insight is that a significant portion of indirect illumination can be visible within the current view frustum (as shown by the connected point pairs in Fig. 1). We propose a novel approach that approximates global illumination through efficient screen-space ray tracing [29]. Specifically, we first perform deferred shading to generate per-pixel G-buffers encoding surface geometry and material properties. Then, we execute a fast Monte-Carlo screen-space ray tracing step directly on these G-buffers to estimate the indirect illumination. Finally, we composite the resulting indirect illumination with direct shading to produce the final rendered image. This method is particularly effective in enclosed environments (e.g. underground garages) where most light-contributing and receiving surfaces are simultaneously visible. Unlike full global illumination methods that require complete scene geometry [6, 32], our screen-space ray tracing approach relies solely on the information available in the current frame, significantly reducing computation latency. Coupled with our customized CUDA kernel, this allows for real-time performance in both novel view synthesis and scene editing.

In summary, we make the following **contributions**:

- We propose an inverse rendering framework for Gaussian Splattings that facilitates approximation of one-bounce global illumination by screen-space ray tracing while preserving real-time scene editability.
- Our framework accurately decomposes scene appearance into intrinsic surface properties and direct/indirect illumination components, enabling realistic edits to geometry, materials, and lighting conditions.
- Our approach delivers realistic global illumination effects while ensuring real-time performance. Our code will be made publicly available.

2. Related Works

Inverse Rendering. Inverse rendering aims to decompose appearance into its intrinsic material and lighting components. It presents a more complex challenge than 3D reconstruction alone, as it is inherently ill-posed - multiple combinations of materials and lighting can produce identical appearances.

Traditional methods tackle this ambiguity through controlled lighting conditions [1, 2, 4, 27, 54]. FIPT [39] introduced a hybrid approach for global illumination with high physical accuracy, IRIS [24] successfully tackled the problem of HDR recovery from LDR inputs with physically-based rendering. Zhu et al. [59] demonstrated impressive results through Monte Carlo raytracing. The field has evolved with data-driven approaches that leverage large-scale models to learn intrinsic property priors [48, 52]. Recent advances in neural representations have transformed inverse rendering approaches. Ref-NeRF [37] introduced Integrated Directional Encoding (IDE) for view-dependent effects, NeILF [46, 53] enabled single-bounce global illumination via incident light field modeling.

The emergence of 3D Gaussian Splatting (3DGS) has led to numerous innovations [13, 36, 40, 49]. GaussianShader [19] links material properties to Gaussians and models appearance through environmental lighting. GS-IR [56] assumes static lighting and caches indirect illumination into a grid of light probes, which limits editability by fixing the indirect radiance in local light maps. Relightable3DGS [14] enables Monte-Carlo ray tracing through a bounding volume hierarchy (BVH) of Gaussians but requires costly BVH

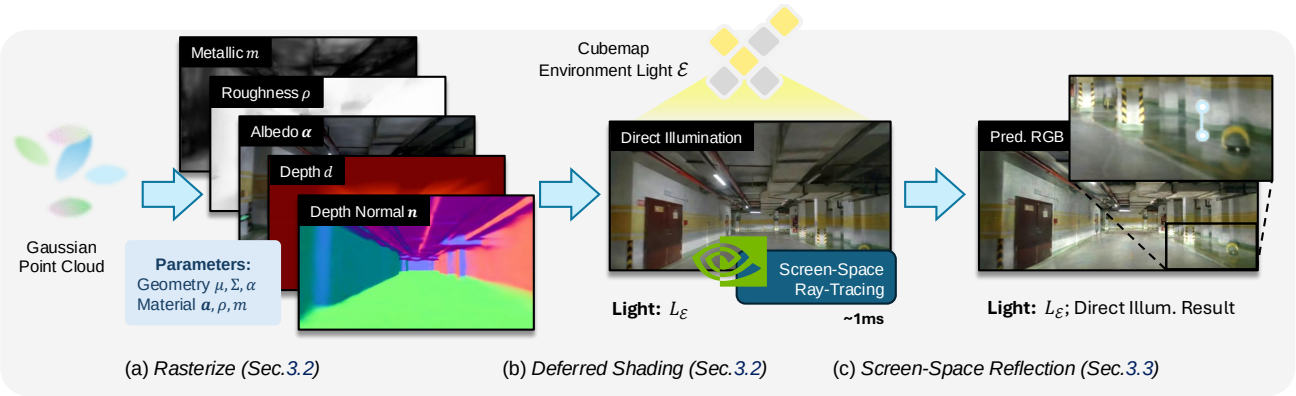

Figure 2. **Pipeline**: (a) Our method extends standard Gaussian splatting with material intrinsic properties. We first rasterize the Gaussian primitives into G-buffers containing both geometric and material properties of the current rendering frame; (b) We perform deferred shading on the alpha-composited G-buffers with direct environment lights from a learnable cubemap. (Sec. 3.2); (c) We approximate one-bounce indirect lights through screen-space ray tracing and compose the final rendered RGB through Monte-Carlo integration. (Sec.3.3).

updates for any geometry modifications. PRTGS [16] encodes light transport in spherical harmonics (SH) parameters but still relies on expensive pre-computation during editing. In contrast, our method computes screen-space reflection dynamically, providing greater flexibility for modeling inter-reflections. Concurrently, GI-GS [8] effectively leveraged screen-space techniques for efficient global illumination calculation.

Neural Simulation. Neural simulation has become a vital technology that connects real-world perception with synthetic training environments for computer vision [15] and robotics [11, 28, 41, 57]. The Real2Sim pipeline facilitates cost-effective training of autonomous systems by creating digital twins of real environments. However, this is particularly challenging in driving scenarios due to the need to accurately model complex reflective surfaces, dynamic motion, sparse viewpoints, and various environmental factors, all while ensuring real-time editing capability.

Recent advances in driving simulation have primarily focused on modeling traffic agent behavior and scene dynamics through neural scene graphs [10, 34, 41, 43, 58]. It was first used by Ost et al. [34], who proposed separating the representation of dynamic vehicles from static backgrounds using distinct neural rendering models. This approach was further developed in UniSim [43] and MARS [38], which incorporated efficient large-scale neural radiance fields [33] to improve rendering quality and computational performance. OmniRe [10] extended these capabilities to handle general dynamic objects [45] and pedestrians, addressing a significant limitation in previous frameworks.

However, these approaches share a common limitation: they typically encode lighting and material properties implicitly within their neural representations. This implicit encoding complicates the modification of scene illumination and the reuse of assets across different lighting conditions. Our work addresses this gap by explicitly modeling material properties and enabling dynamic global illumination, supporting more flexible simulation environments.

3. Methods

As illustrated in Fig. 2, our method extends Gaussian Splatting with physically-based materials and efficient screen-space global illumination. In Sec. 3.1, we review the fundamentals of physically-based rendering and the Gaussian Splatting framework. We then present our deferred shading pipeline in Sec. 3.2, which efficiently processes the rasterized G-buffers to compute direct illumination. In Sec. 3.3, we detail our Monte-Carlo screen-space ray tracing technique that approximates indirect illumination. Finally, we describe our optimization framework in Sec. 3.4.

3.1. Preliminaries

The Rendering Equation. The rendering equation [21] describes the light transport at any surface point p. The outgoing radiance $\mathbf{c}$ in direction $\boldsymbol{\omega_o}$ is the sum of reflected incident lights from all directions:

$$\mathbf{c}(\boldsymbol{\omega_o}) = \int_{\Omega^+} L_i(\boldsymbol{\omega_i}) f(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})(\boldsymbol{\omega_i} \cdot \mathbf{n})\, \mathrm{d}\boldsymbol{\omega_i}\,, \qquad (1)$$

where $L_i(\boldsymbol{\omega_i})$ is the incident radiance from direction $\boldsymbol{\omega_i}$ over the hemisphere Ω^+, f is the Bidirectional Reflectance Distribution Function (BRDF), and $\boldsymbol{\omega_i} \cdot \mathbf{n}$ accounts for the Lambert's cosine law. Unlike the original rendering equation [21], we omit the emission term $\mathbf{c}_e(\boldsymbol{\omega_o})$. This design choice stems from our observation that emission terms can dominate the produced radiance during optimization, effectively suppressing proper material decomposition. Including such terms risks degrading our inverse rendering pipeline to behave like the original Gaussian Splatting, which lacks relighting capabilities.

3D Gaussian Rasterization. 3D Gaussian Splatting (3DGS) [5] represents a scene as a set of 3D Gaussian primitives that can be efficiently rasterized through alpha-blending. Each Gaussian is parameterized by its spatial position $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$, and appearance attributes including opacity α and spherical harmonics coefficients for view-dependent color. The 3D Gaussian distribution is defined as:

$$G(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))\,. \tag{2}$$

For efficient rendering, the covariance matrix is decomposed into rotation and scale components $\boldsymbol{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top$, where $\mathbf{R}$ is a rotation matrix and $\mathbf{S}$ is a scaling matrix.

During rendering, these 3D Gaussians are projected to 2D screen space and alpha-blended in front-to-back order. The final pixel color $\mathbf{C}$ is computed as:

$$\mathbf{C} = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i\,, \tag{3}$$

where $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$ is the accumulated transmittance, α_i is the opacity, and $\mathbf{c}_i$ is the view-dependent color of the i-th Gaussian.

While this rasterization-based approach enables efficient rendering, it presents challenges for global illumination simulation. The key limitation stems from its **forward-only accumulation nature** – each Gaussian can only receive illumination from directly visible light sources, making it difficult to model indirect bounces that require tracking light paths through multiple surfaces.

3.2. Deferred Shading

To enable physically-based rendering, we extend each Gaussian with material properties following Disney's principled BRDF [7]. Specifically, each Gaussian is augmented with a diffuse albedo $\mathbf{a} \in [0,1]^3$, roughness $\rho \in [0,1]$, and metallic parameter $m \in [0,1]$. Following previous works [40, 49], we employ a deferred rendering process that separates the material composition and shading stages. We first alpha-composites the material parameters into 2D buffers (i.e. G-Buffers, see Fig. 2-a) and then perform the shading computations on these buffers (Fig. 2-b). Compared to the per-Gaussian shading process, this saves unnecessary computation on the Gaussians that are invisible to the current viewing frustum and prevents blending artifacts as stated in [40].

Similar to Eq. (3), the material properties are alpha-composited into G-buffers as:

$$G_p = \sum_{i=1}^{N} T_i \alpha_i p_i\,, \tag{4}$$

where $p \in \{\mathbf{a}, \rho, m, \gamma\}$ represents the material property.

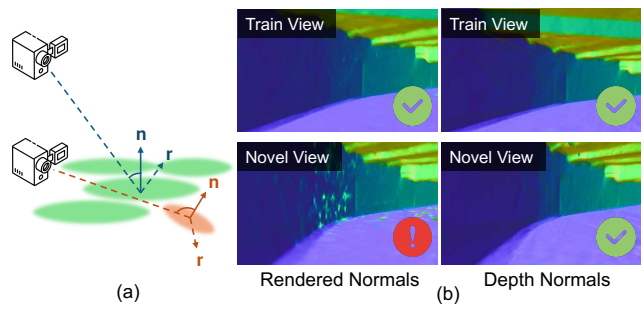

Figure 3. (a) While alpha-composited normals work well from the original camera view (top), they can produce incorrect estimates in novel views (bottom) when rays intersect with previously occluded surfaces. (b) Qualitative comparison showing that depth-based normals maintain consistency across both training and novel views, while rendered normals exhibit artifacts in novel views.

Different from prior inverse rendering approaches [19, 56] that directly rasterize per-Gaussian normals, we compute surface normals from the rendered depth buffer (dubbed depth normal), which we found to be more robust in practice. Similar to 2DGS [18], the depth normal is computed through finite differences.

Per-Gaussian v.s. Depth Normal. While previous methods [19, 56] obtain surface normals through the alpha composition of per-Gaussian normals (Rendered Normals), we find that this can lead to inconsistent normal estimates at novel views (Fig. 3-b, bottom left). This is because that the rendered normals only represent the orientation of **individual** Gaussians along each ray, without considering the underlying local geometry structure. In contrast, depth-based normal estimation leverages the spatial distribution of reconstructed surface points to compute geometrically meaningful normals (Fig. 3-b, right column). The superior robustness of depth normals is crucial for stable view synthesis, as accurate surface normals are essential for the shading process. Please find more details in the supplementary.

Physically-Based BRDF. We adopt a simplified Disney BRDF [7] that models the surface interaction through a combination of diffuse and specular terms, parameterized by diffuse albedo $\mathbf{a}$, roughness ρ, and metallic m:

$$f = \underbrace{\frac{1-m}{\pi}\mathbf{a}}_{f_d} + \underbrace{\frac{DFG}{4(\mathbf{n}\cdot\boldsymbol{\omega_i})(\mathbf{n}\cdot\boldsymbol{\omega_o})}}_{f_s}\,, \tag{5}$$

where D, F, and G represent the normal distribution, Fresnel, and geometry terms respectively. The equations for these terms are detailed in the supplementary materials.

Combining the BRDF with the rendering equation (Eq. (1)), the outgoing radiance of a shading point can be

expressed as $\mathbf{c} = \mathbf{c}_d + \mathbf{c}_s$, where:

$$\mathbf{c}_d = \frac{1-m}{\pi}\mathbf{a}\int_{\Omega+} L_i(\boldsymbol{\omega_i})(\boldsymbol{\omega_i}\cdot\mathbf{n})\,\mathrm{d}\boldsymbol{\omega_i}\,, \tag{6}$$

$$\mathbf{c}_s = \int_{\Omega+} L_i(\boldsymbol{\omega_i}) f_s(\boldsymbol{\omega_i}\cdot\mathbf{n})\,\mathrm{d}\boldsymbol{\omega_i}\,. \tag{7}$$

A direct solution to these integrals would require considering all possible light interactions between points in the scene, resulting in an algorithm with $\mathcal{O}(N^2)$ time complexity (N being the number of points). This approach is computationally prohibitive in practice. To make physically-based rendering feasible with 3DGS, we employ the split-sum approximation [22] (detailed below), which allows us to consider only the dominant incident light direction. The approximations reduce the computational complexity to linear time while maintaining high rendering quality.

Shading. With the G-buffers providing the necessary information, we compute the first-pass shading that captures direct illumination from the environment. (Fig. 2-b).

As solving the full integrals in Eq. (6) and (7) is computationally prohibitive, we employ the split-sum approximation [22] that separates the "sum of product" integral, where lighting and BRDF are entangled, into the product of two integrals. For the specular term $\mathbf{c}_s$, this gives:

$$\mathbf{c}_s \approx \underbrace{\int_{\Omega+} L_i(\boldsymbol{\omega_i}) D(\rho, \mathbf{r})\,\mathrm{d}\boldsymbol{\omega_i}}_{\text{specular light integral}} \cdot \underbrace{\int_{\Omega+} f_s\cdot(\boldsymbol{\omega_i}\cdot\mathbf{n})\,\mathrm{d}\boldsymbol{\omega_i}}_{\text{BRDF integral } F_s}\,, \tag{8}$$

where $\mathbf{r}$ is the reflection direction $\mathbf{r}$ calculated from the view direction $\mathbf{v}$ and surface normal $\mathbf{n}$ as $\mathbf{r} = 2(\mathbf{v}\cdot\mathbf{n})\mathbf{n} - \mathbf{v}$.

The BRDF integral can be pre-computed and stored in a 2D lookup table indexed by roughness ρ and the cosine angle of $\mathbf{n}\cdot\mathbf{v}$. Detailed explanations are provided in the supplementary materials.

The specular light integral represents the color of the reflection lobe, whose direction follows $\mathbf{r}$ and whose width is determined by the surface roughness ρ - rougher surfaces receive a broader range of reflection. We implement this through a range query on the environment light using a learnable cubemap $\mathcal{E}$ with the NVDiffrast [23] library. This process is expressed as:

$$L_{\mathcal{E}}(\mathbf{r}, G_\rho) = \mathrm{SampleEnvMap}(\mathbf{r}, \lambda, \mathcal{E})\,, \tag{9}$$

where the mipmap level λ is computed empirically as $\lambda = \log_2(G_\rho + 1)\lambda_{\max}$, with $\lambda_{\max}$ being the maximum mipmap level. This ensures proper integration over the specular lobe defined by the material roughness.

With these approximations, the specular term $\mathbf{c}_s$ can be simplified to:

$$\mathbf{c}_s = L_{\mathcal{E}}(\mathbf{r}, G_\rho)\cdot F_s, \tag{10}$$

where G_ρ is the alpha-composited roughness buffer as in Eq. (4). The diffuse color $\mathbf{c}_d$ can be simplified similarly by querying the environment light map on the surface normal direction as $\mathbf{c}_d = (1 - G_m)G_{\mathbf{a}} \cdot L_{\mathcal{E}}(\mathbf{n}, G_\rho)$. The first-pass radiance $\mathbf{c}_1$ can then be obtained by combining the diffuse and specular components: $\mathbf{c}_1 = \mathbf{c}_d + \mathbf{c}_s$.

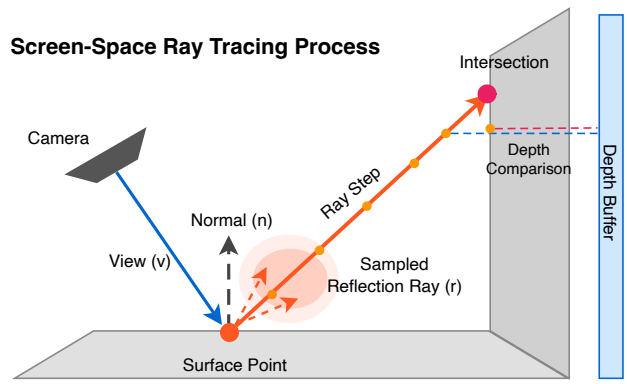

Figure 4. Screen-space ray tracing process. For each pixel, we march along the sampled reflection ray step by step and compare the ray's depth with the scene depth buffer at each step. An intersection is detected when the ray depth transitions from being in front of to behind the depth buffer values, indicating the ray has intersected with scene geometry.

3.3. Screen-space Ray Tracing

While the first-pass shading captures direct illumination, it fails to account for indirect bounces that are crucial for realistic rendering, particularly in indoor environments. We propose to incorporate an efficient screen-space ray tracing approach that approximates one-bounce indirect illumination by leveraging the information available in the G-buffers [29] (Fig. 2-c).

We illustrate the screen-space tracing process in Fig. 4. For each pixel, we trace reflection rays in screen space to find potential indirect light contributions (shown as corresponding point pairs in Fig. 2-c). Given a pixel's world position $\mathbf{p}$ (un-projected from the depth buffer), view direction $\mathbf{v}$, and normal $\mathbf{n}$, we generate a reflection ray as $\mathbf{r}(t)$ with t being the marched distance.

To detect intersections, we project the ray into screen space and march along it in fixed steps. At each step i, we compare the ray's depth z_{ray} with the scene depth z_{scene} from the depth buffer:

$$\Delta z_i = z_{\text{ray}}(t_i) - z_{\text{scene}}\big(\mathrm{proj}\big(\mathbf{r}(t_i)\big)\big)\,, \tag{11}$$

where proj denotes perspective projection. An intersection is detected when Δz_i changes from negative to positive, indicating the ray has passed through a surface. We denote the UV coordinate of the intersection point as p^*.

In practice, we sample $N_s = 8$ reflection rays from the reflection lobe and trace the reflection colors $\{\mathbf{c}_{1,i}\}_{N_s}$ in the screen space. Then, the sampled colors are combined using

		Garage-0			Garage-1			Garage-2			Garage-3			Campus-0			Campus-1		
		PSNR*	SSIM*	LPIPS	PSNR*	SSIM*	LPIPS	PSNR*	SSIM*	LPIPS	PSNR*	SSIM*	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
NVS	PVG [9]	28.41	0.8415	0.1645	26.99	0.6926	0.2721	30.71	0.8486	0.2355	34.68	0.9407	0.1211	28.36	0.8408	0.2562	26.40	0.8792	0.1881
	StreetGS [42]	26.60	0.8172	0.1716	25.52	0.6245	0.3395	27.77	0.7811	0.2844	36.39	0.9598	0.1409	29.07	0.8627	0.2018	26.54	0.8888	0.1239
	OmniRe [10]	26.83	0.8227	0.1712	25.72	0.6291	0.3370	28.25	0.7965	0.2818	36.04	0.9586	0.1402	29.38	0.8631	0.2018	26.53	0.8889	0.1241
	GShader [19]	25.02	0.7343	0.2837	23.70	0.5334	0.5060	24.56	0.6571	0.4259	33.88	0.9029	0.1530	24.57	0.7821	0.4463	25.04	0.8646	0.1755
	Ours	28.14	0.8374	0.1587	25.99	0.6609	0.1694	27.72	0.7816	0.2295	36.23	0.9583	0.0424	29.66	0.8646	0.1812	26.37	0.8704	0.1389
Recon	PVG [9]	28.90	0.7986	0.1621	28.89	0.7987	0.2940	32.20	0.8842	0.2291	38.47	0.9631	0.1367	30.19	0.8749	0.2329	28.15	0.9017	0.1750
	StreetGS [42]	27.78	0.8537	0.1655	27.46	0.7764	0.2655	28.92	0.8229	0.2776	38.23	0.9646	0.1499	30.88	0.8941	0.1919	29.12	0.9223	0.1126
	OmniRe [10]	27.30	0.7746	0.1635	27.30	0.7746	0.2689	29.63	0.8436	0.2747	38.43	0.9691	0.1413	30.86	0.8960	0.1925	29.15	0.9224	0.1126
	GShader [19]	26.57	0.7965	0.272	23.89	0.5608	0.5025	27.01	0.7706	0.2867	36.54	0.9606	0.1792	25.03	0.7920	0.4388	26.90	0.8957	0.1657
	Ours	28.82	0.8720	0.1348	28.47	0.8232	0.1591	29.16	0.8288	0.1945	36.97	0.9562	0.0458	32.02	0.9066	0.1610	29.07	0.9115	0.1205

Table 1. Quantitative evaluation results. * For PSNR and SSIM computation, we exclude the region that contains the ego car in the image.

Monte-Carlo integration:

$$\mathbf{c}'_s = \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{f_s \mathbf{c}_{1,i}(\omega_i \cdot n)}{p_{GGX}(\omega_i)}, \tag{12}$$

where ω_i is the i-th reflection direction sampled from the GGX [38] distribution p_{GGX}.

The final rendered color is obtained by combining the diffuse component with the Monte Carlo integrated specular reflection, followed by tone mapping and gamma correction to convert the result into the standard RGB color space.

3.4. Optimization

Our optimization objective combines several losses to ensure high-quality reconstruction of both geometry and material properties. The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{o}}\mathcal{L}_{\text{opacity}} + \lambda_{\text{n}}\mathcal{L}_{\text{n}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}\,, \tag{13}$$

where $\mathcal{L}_{\text{rgb}}$ is the standard RGB reconstruction loss, $\mathcal{L}_{\text{opacity}}$ encourages the Gaussians to fully cover the images, $\mathcal{L}_{\text{n}}$ is a depth-normal consistency loss following [18], and $\mathcal{L}_{\text{reg}}$ is a total variance loss applied on the G-buffers. Details on the loss functions are provided in the supplementary.

4. Experiments

Baselines. We compare our method with 3 lines of baseline methods: 1) Dynamic view synthesis algorithm: We select Periodic Vibration Gaussian (PVG) [9], which models appearance changes through temporal-dependent Gaussians. 2) Driving Simulators: We compare against Street Gaussians (StreetGS) [42], which augments 3D Gaussians with dynamic spherical harmonics for appearance changes. We also compare with OmniRe [10], which constructs dynamic neural scene graphs using multiple types of dynamic nodes to model various kinds of actors in driving scenes. 3) Inverse rendering algorithm: We include GaussianShader (GShader) [19] as our primary baseline for inverse rendering comparison, as it also builds upon Gaussian Splatting but only uses cubemap lighting.

To ensure fair comparisons, all methods are initialized with identical point clouds derived from LiDAR captures. We maintain the original hyperparameters as specified in each method's paper. For PVG [9] and StreetGS [42], we utilize the implementations provided in the OmniRe repository [10]. GShader [19] is integrated directly into our evaluation pipeline through code migration to ensure consistent evaluation conditions.

Dataset. Our evaluation uses a self-collected dataset comprising 4 sequences from underground garages and 2 campus scenes. Garage sequences are captured using a multi-sensor setup: three synchronized RGB cameras operating at 30 FPS, accompanied by LiDAR scans for geometric reference. Campus scenes are captured with a hand-held scanner with calibrated LiDAR and images [25]. These environments present challenging lighting conditions with multiple light sources, significant indirect illumination, and various surface materials.

We conduct quantitative experiments under two protocols established by prior works [10, 34, 41]: 1) Scene Reconstruction: Using all frames for both training and testing to evaluate the method's reconstruction fidelity; 2) Novel View Synthesis: Using 50% of frames for training and the remainder for testing, selecting test frames uniformly across the sequence to assess generalization.

4.1. Quantitative Evaluations

Metrics. Due to the presence of ego vehicle parts in our captured images, we modify all baseline methods to exclude those regions in the reconstruction losses during training. Each method is trained for 30,000 steps. We evaluate performance using standard image quality metrics: PSNR, SSIM, and LPIPS [55]. For PSNR and SSIM calculations, we exclude regions containing the ego vehicle to focus on scene reconstruction quality.

Results. Table 1 shows the quantitative comparison results. PVG [9] (first row) serves as an effective upper bound in our quantitative evaluations due to its unique approach to modeling time-dependent Gaussians. While it cannot extrapolate views with physically plausible reflections, it achieves superior reconstruction and view interpolation quality by encoding light-surface interactions as temporal features. This makes it particularly effective at reproducing the training sequence, albeit without the physical

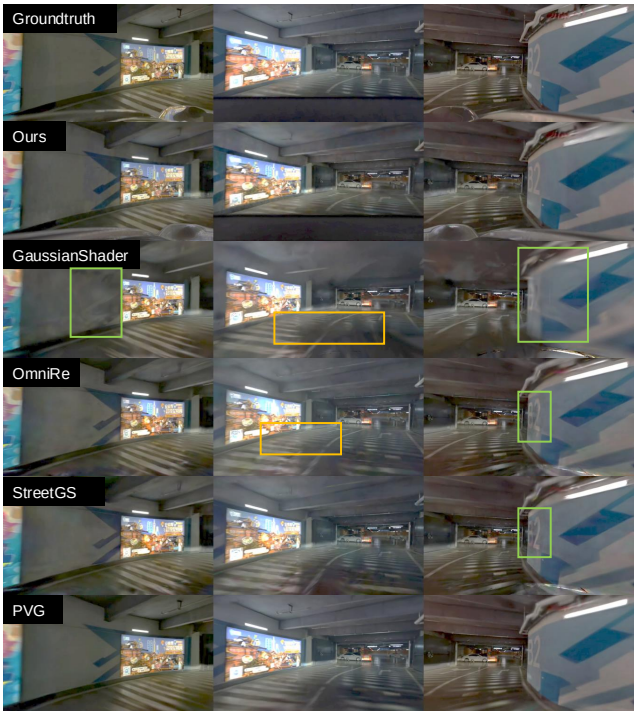

Figure 5. Qualitative comparisons. We show the rendering results for each method. The regions highlighted in yellow boxes show reflections on the ground where baseline methods struggle to capture dramatically dynamic indirect illumination effects; Green boxes show blurry artifacts.

	PVG [9]	StreetGS [42]	OmniRe [10]	GShader [19]	Ours w/o SSR	Ours
FPS	72	69	71	74	38	37

Table 2. Real-time rendering performance comparison. FPS is measured using an NVIDIA 4090 at 960×540 resolution.

interpretability that our method provides.

Beyond the comparison with PVG, our method is on par with other 3D reconstruction baselines (row 2 and 3) in both settings and consistently outperforms the inverse rendering baseline GShader [19] (fourth row), which lacks enough capacity to model indirect lighting effects.

Rendering speed. We evaluate the real-time rendering performance of our method against the baselines. As shown in Table 2, reconstruction-based methods like PVG [9], StreetGS [42], OmniRe [10], and GShader [19] achieve 69-74 FPS. Our method maintains 37 FPS while computing physically-based shading and global illumination effects. The small overhead between our full method and the variant without screen-space reflections (38 vs 37 FPS) demonstrates that our ray marching implementation adds minimal computational cost.

4.2. Qualitative Evaluation

We demonstrate our method's capabilities through several visual experiments:

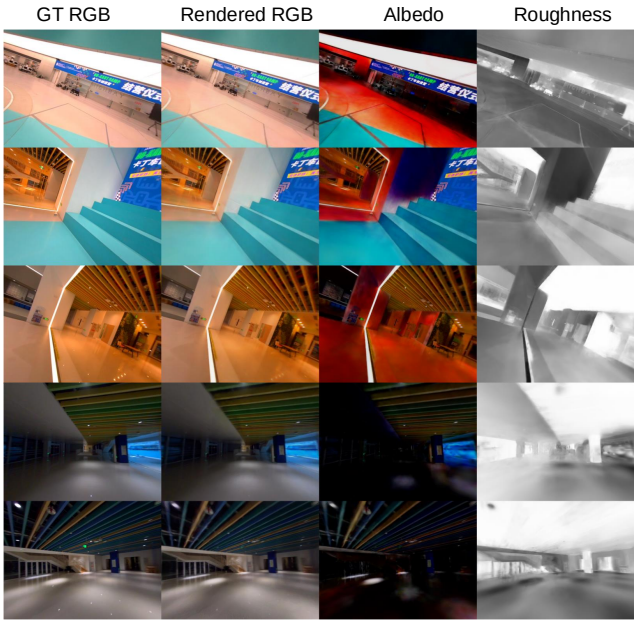

Figure 6. Decomposition results. We show our decomposed intrinsic channels. The contrast ratio of the consistency channels is amplified for visualization.

Reconstruction. As shown in Fig. 5, our method achieves faithful reconstruction quality compared to baseline approaches, particularly in handling reflective surfaces and indirect illumination effects. The most notable differences appear in the reflective floor regions (highlighted in yellow boxes) where GShader [19] exhibits significant artifacts due to its limited modeling of global illumination. Standard reconstruction methods like OmniRe [10] and StreetGS [42] also struggle with these challenging lighting scenarios, producing blurry or inconsistent reflections (highlighted in green boxes). In contrast, our physically-based shading model successfully captures the complex interplay of light, accurately reproducing both the billboard reflections on the floor and the subtle ambient lighting throughout the parking garage. PVG [9] achieves comparable visual quality but does so through temporal encoding rather than explicit physical modeling.

Decomposition. Our method achieves convincing decomposition of scene appearance into its constituent components. As shown in Fig. 6, we can separate the complex visual effects in indoor scenes into intrinsic components. The albedo map reveals the intrinsic surface colors without the influence of illumination. We also show decomposition results of small objects from TensoIR [20] dataset in Fig. 7.

Relighting. As demonstrated in Fig. 7, our method successfully handles various lighting conditions by incorporating HDRI environment maps from online sources [17, 20, 35].

Object Insertion. We demonstrate the seamless integration of external 3D assets into the reconstructed scene. The pro-

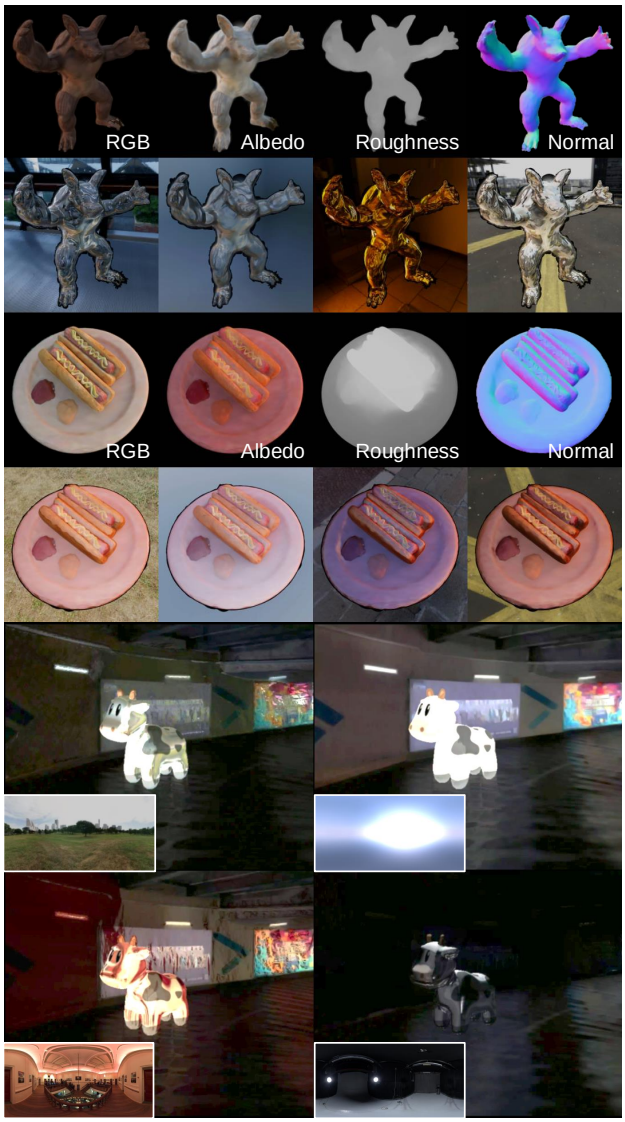

Figure 7. Top: Intrinsic channels and editing results (relighting, material editing) from NeRF-Synthetic dataset [30]. Bottom: Relighted garage scene with inserted object.

cess begins by rendering the inserted object into G-buffers using NVDiffrast [23]. These buffers are then composited with the scene's G-buffers through depth-based comparison. As shown in Fig. 7, this approach enables the inserted object to receive proper illumination while contributing to indirect lighting effects, evidenced by the realistic reflections on the floor.

Material Editing. Our method enables direct manipulation of material properties, including roughness and metallic parameters (Fig. 7). By adjusting these physically-based parameters, we can achieve a range of surface appearances from diffuse to highly specular, while maintaining consistent interactions with both direct and indirect illumination.

Our decomposed representation also allows for intuitive editing of surface colors. As demonstrated in Fig. 8, we

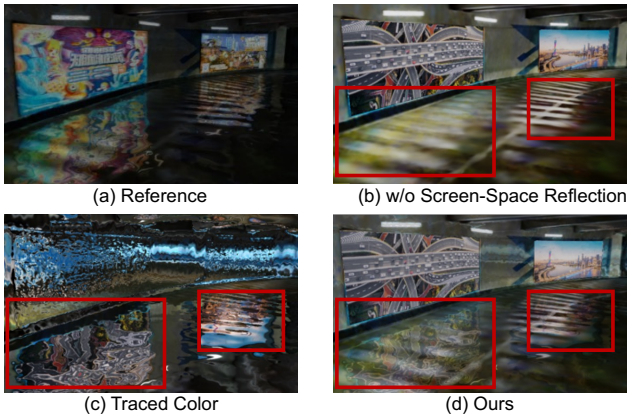

Figure 8. Effect of screen-space reflections during editing. (a) Reference image. (b) Without modeling screen-space reflections, the floor reflections remain unchanged after editing since they are baked into the albedo. (c) Visualization of the screen-space traced lights. (d) Our method reflects the edited billboard content. We set roughness to a relatively low value to exaggerate the effect.

can easily modify scene content by editing the albedo channel. Our method automatically updates all lighting interactions to maintain physical consistency, with floor reflections dynamically adapting to the new content. This showcases how our approach preserves the natural interaction between edited materials and global illumination.

Effect of Screen-Space Reflections. Fig. 8 also serves to demonstrate the importance of our screen-space reflection technique. Without it, reflections become baked into the ground floor albedo thus showing noticeable artifacts at novel views (b). Our method properly handles indirect illumination through ray tracing, ensuring reflections update accurately with edited content (d).

5. Conclusion

We have developed a novel inverse rendering framework for Gaussian Splatting that enables real-time editing with global illumination effects. By combining screen-space ray tracing with Gaussian representations and a learned consistency parameter, our method achieves physically plausible rendering and interactive performance. It allows for scene decomposition into editable components, facilitating operations like object insertion, material editing, and relighting in complex indoor environments. Experimental results confirm the effectiveness of our approach and realistic screen-space reflections. However, our method has limitations. The non-differentiable nature of screen-space ray marching can impact optimization stability, suggesting a need for numerical gradient techniques. Our method's performance decreases in outdoor settings with distant light interactions, indicating a need for a hybrid approach to managing varying scales of light transport.

References

[1] Jens Ackermann, Michael Goesele, et al. A survey of photometric stereo techniques. *Foundations and Trends® in Computer Graphics and Vision*, 9(3-4):149–254, 2015. 2

[2] Jonathan T. Barron and Jitendra Malik. Shape, Illumination, and Reflectance from Shading. *TPAMI 2015*, Oct. 2020. 2

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, Oct. 2021. 1

[4] Harry Barrow and J. Tenenbaum. Recovering intrinsic scene characteristics from images. *Computer Vision System*, 2:3–26, Jan. 1978. 2

[5] Kerbl Bernhard, Kopanas Georgios, Leimkühler Thomas, and Drettakis George. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, May 2023. 1, 2, 4

[6] Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. RayGauss: Volumetric Gaussian-Based Ray Casting for Photorealistic Novel View Synthesis, Aug. 2024. 2

[7] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *ACM SIGGRAPH*, volume 2012, pages 1–7. vol. 2012, 2012. 4

[8] Hongze Chen, Zehong Lin, and Jun Zhang. GI-GS: GLOBAL ILLUMINATION DECOMPOSITION ON GAUSSIAN SPLATTING FOR INVERSE RENDERING. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[9] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic Vibration Gaussian: Dynamic Urban Scene Reconstruction and Real-time Rendering, Mar. 2024. 6, 7

[10] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, Li Song, and Yue Wang. OmniRe: Omni Urban Scene Reconstruction. In *The Thirteenth International Conference on Learning Representations*, Aug. 2024. 3, 6, 7

[11] Sammy Christen, Lan Feng, Wei Yang, Yu-Wei Chao, Otmar Hilliges, and Jie Song. Synh2r: Synthesizing hand-object motions for learning human-to-robot handovers. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3168–3175, 2024. 3

[12] Blender Online Community. *Blender - a 3D Modelling and Rendering Package*. Stichting Blender Foundation, Amsterdam, 2018. 1

[13] Kang Du, Zhihao Liang, and Zeyu Wang. GS-ID: Illumination Decomposition on Gaussian Splatting via Diffusion Prior and Parametric Light Source Optimization, Aug. 2024. 2

[14] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing. In *European Conference on Computer Vision*, 2024. 2

[15] Chen Guo, Tianjian Jiang, Xu Chen, Jie Song, and Otmar Hilliges. Vid2Avatar: 3D Avatar Reconstruction from Videos in the Wild via Self-supervised Scene Decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12858–12868, Canada, Feb. 2023. IEEE. 3

[16] Yijia Guo, Yuanxi Bai, Liwen Hu, Ziyi Guo, Mianzhi Liu, Yu Cai, Tiejun Huang, and Lei Ma. PRTGS: Precomputed Radiance Transfer of Gaussian Splats for Real-Time High-Quality Relighting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, pages 5112–5120, New York, NY, USA, Oct. 2024. Association for Computing Machinery. 2, 3

[17] HDRI Haven. Hdri haven. 7

[18] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 1, 4, 6

[19] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nov. 2023. 2, 4, 6, 7

[20] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. TensoIR: Tensorial Inverse Rendering, Mar. 2024. 7

[21] James T. Kajiya. The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20(4):143–150, Aug. 1986. 3

[22] Brian Karis. Real Shading in Unreal Engine 4. *SIGGRAPH Tutorial*, 2013. 5

[23] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular Primitives for High-Performance Differentiable Rendering. In *ACM Transactions on Graphics (ToG)*, Nov. 2020. 5, 8

[24] Chih-Hao Lin, Jia-Bin Huang, Zhengqin Li, Zhao Dong, Christian Richardt, Tuotuo Li, Michael Zollhöfer, Johannes Kopf, Shenlong Wang, and Changil Kim. IRIS: Inverse Rendering of Indoor Scenes from Low Dynamic Range Images, Jan. 2025. 2

[25] Bonan Liu, Guoyang Zhao, Jianhao Jiao, Guang Cai, Chengyang Li, Handi Yin, Yuyang Wang, Ming Liu, and Pan Hui. OmniColor: A Global Camera Pose Optimization Approach of LiDAR-360Camera Fusion for Colorizing Point Clouds. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Apr. 2024. 6

[26] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. NeRO: Neural geometry and BRDF reconstruction of reflective objects from multiview images. *ACM Trans. Graph.*, 42(4), July 2023. 2

[27] Stephen Lombardi and Ko Nishino. Reflectance and Illumination Recovery in the Wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):129–141, Jan. 2016. 2

[28] Haozhe Lou, Yurong Liu, Yike Pan, Yiran Geng, Jianteng Chen, Wenlong Ma, Chenglong Li, Lin Wang, Hengzhen Feng, Lu Shi, Liyi Luo, and Yongliang Shi. Robo-GS: A Physics Consistent Spatial-Temporal Model for Robotic Arm with Hybrid Representation, Aug. 2024. 3

[29] Morgan McGuire and Michael Mara. Efficient GPU screen-space ray tracing. *Journal of Computer Graphics Techniques*

(JCGT), 3(4):73–85, Dec. 2014. 2, 5

[30] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul Srinivasan, and Jonathan T. Barron. NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images, Nov. 2021. 8

[31] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 405–421, Cham, 2020. Springer International Publishing. 1

[32] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *SIGGRAPH (Asia) 2024*, July 2024. 2

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. 1, 3

[34] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural Scene Graphs for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Mar. 2021. 3, 6

[35] Emil Persson. Humus. 7

[36] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jingtuo Liu, Liangjun Zhang, Jian Zhang, Bin Zhou, Errui Ding, and Jingdong Wang. GIR: 3D Gaussian Inverse Rendering for Relightable Scene Factorization, Dec. 2023. 2

[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. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. 2

[38] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 195–206, Goslar, DEU, June 2007. Eurographics Association. 3, 6

[39] Liwen Wu, Rui Zhu, Mustafa B. Yaldiz, Yinhao Zhu, Hong Cai, Janarbek Matai, Fatih Porikli, Tzu-Mao Li, Manmohan Chandraker, and Ravi Ramamoorthi. Factorized Inverse Path Tracing for Efficient and Accurate Material-Lighting Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3848–3858, 2023. 2

[40] Tong Wu, Jia-Mu Sun, Yu-Kun Lai, Yuewen Ma, Leif Kobbelt, and Lin Gao. DeferredGS: Decoupled and Editable Gaussian Splatting with Deferred Shading, May 2024. 2, 4

[41] Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuantao Chen, Runyi Yang, Yuxin Huang, Xiaoyu Ye, Zike Yan, Yongliang Shi, Yiyi Liao, and Hao Zhao. MARS: An Instance-aware, Modular and Realistic Simulator for Autonomous Driving. In *CAAI International Conference on Artificial Intelligence (CICAI)*, July 2023. 3, 6

[42] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street Gaussians for Modeling Dynamic Urban Scenes. In *European Conference On Computer Vision*. IEEE, Jan. 2024. 6, 7

[43] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. UniSim: A Neural Closed-Loop Sensor Simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023. 3

[44] Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-Gaussian: Anisotropic View-Dependent Appearance for 3D Gaussian Splatting, Feb. 2024. 2

[45] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. In *Computer Vision And Pattern Recognition*. IEEE, Sept. 2023. 3

[46] Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. NeILF: Neural Incident Light Field for Physically-based Material Estimation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, volume 13691, pages 700–716. Springer Nature Switzerland, Cham, 2022. 2

[47] Chongjie Ye, Yinyu Nie, Jiahao Chang, Yuantao Chen, Yihao Zhi, and Xiaoguang Han. GauStudio: A Modular Framework for 3D Gaussian Splatting and Beyond, Mar. 2024. 1

[48] Chongjie Ye, Lingteng Qiu, Xiaodong Gu, Qi Zuo, Yushuang Wu, Zilong Dong, Liefeng Bo, Yuliang Xiu, and Xiaoguang Han. StableNormal: Reducing Diffusion Variance for Stable and Sharp Normal. In *SIGGRAPH Asia 2024*. SIGGRAPH Asia, June 2024. 2

[49] Keyang Ye, Qiming Hou, and Kun Zhou. 3D Gaussian Splatting with Deferred Reflection. In *ACM SIGGRAPH 2024*, June 2024. 2, 4

[50] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *Computer Vision And Pattern Recognition*. IEEE, Nov. 2023. 1

[51] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting. In *Computer Vision And Pattern Recognition*. arXiv, Dec. 2023. 1

[52] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. RGB↔X: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24*, pages 1–11, Denver CO USA, July 2024. ACM. 2

[53] Jingyang Zhang, Yao Yao, Shiwei Li, Jingbo Liu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. NeILF++: Inter-Reflectable Light Fields for Geometry and Material Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3601–3610, 2023. 2

[54] Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. IRON: Inverse Rendering by Optimizing Neural SDFs and

Materials from Photometric Images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5555–5564, New Orleans, LA, USA, June 2022. IEEE. 2

[55] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, June 2018. 6

[56] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. GS-IR: 3D Gaussian Splatting for Inverse Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Dec. 2023. 2, 4

[57] Zhide Zhong, Jiakai Cao, Songen Gu, Sirui Xie, Liyi Luo, Hao Zhao, Guyue Zhou, Haoang Li, and Zike Yan. Structured-nerf: Hierarchical scene graph with neural representation. In *European Conference on Computer Vision*, pages 184–201. Springer, 2025. 3

[58] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. HUGS: Holistic Urban 3D Scene Understanding via Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Mar. 2024. 3

[59] Jingsen Zhu, Fujun Luan, Yuchi Huo, Zihao Lin, Zhihua Zhong, Dianbing Xi, Rui Wang, Hujun Bao, Jiaxiang Zheng, and Rui Tang. Learning-based inverse rendering of complex indoor scenes with differentiable monte carlo raytracing. In *SIGGRAPH Asia 2022 Conference Papers*. ACM, 2022. 2

[60] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, and Marc Pollefeys. NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM. In *3DV*. 3DV, Feb. 2023. 1