CG-IR: Curved Gaussian Splatting for Inverse Rendering

Hanxiao Sun[1] Yupeng Gao[2] Jin Xie[2] Jian Yang[2]
Beibei Wang[2,*]

[1]Nankai University [2]Nanjing University

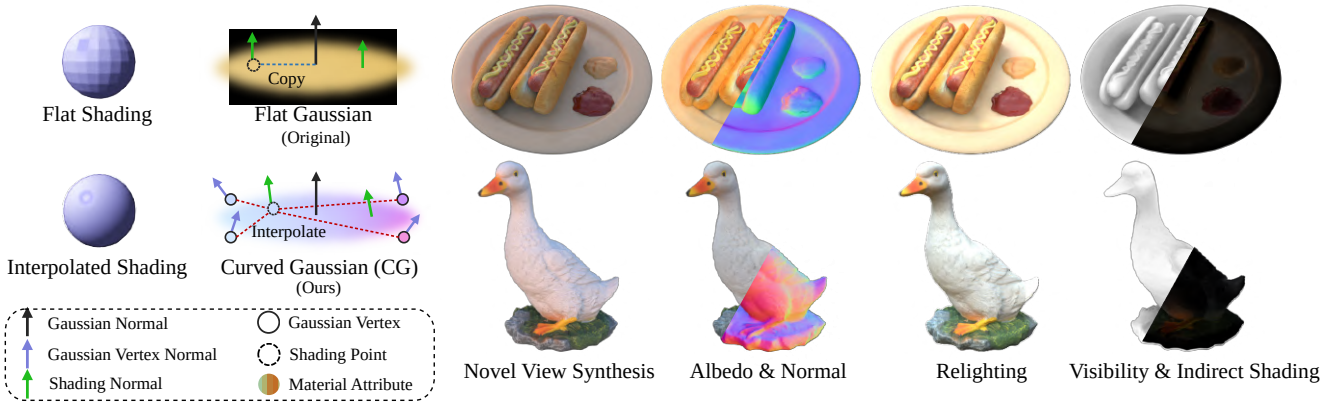


Figure 1. CG-IR introduces a new curved Gaussian representation inspired by replacing flat shading for interpolated shading in triangle rendering. Each curved Gaussian allows spatially-varying material and normal attributes by interpolating among Gaussian vertices defined in the tangent space of a Gaussian. Compared to the original Gaussian (i.e., Flat Gaussian) with constant attributes, CG has a more powerful representation ability to produce high-quality NVS and relighting results.

Abstract

Reconstructing 3D assets from images, known as inverse rendering (IR), remains a challenging task due to its ill-posed nature and the complexities of appearance and lighting. 3D Gaussian Splatting (3DGS) has demonstrated impressive capabilities for novel view synthesis (NVS) tasks. It has also been introduced into relighting by decoupling radiance into Bidirectional Reflectance Distribution Function (BRDF) parameters and environmental lighting. Unfortunately, these methods often produce inferior relighting quality, exhibiting visible artifacts and unnatural indirect illumination. The main reason is the limited capability of each Gaussian, which has constant material parameters and normal, alongside the absence of physical constraints for indirect lighting. In this paper, we present a novel framework called Curved Gaussian Inverse Rendering (CG-IR), aimed at enhancing both NVS and relighting quality. To this end, we propose a new representation—Curved Gaussian (CG)—that generalizes per-Gaussian constant material parameters to allow for spatially varying parameters, indicating that different regions of each Gaussian can have various normals and material properties. This enhanced representation is complemented by a CG splatting scheme akin to vertex/fragment shading in traditional graphics pipelines. Furthermore, we integrate a physically-based indirect lighting model, enabling more realistic relighting. The proposed CG-IR framework significantly improves rendering quality, outperforming state-of-the-art NeRF-based methods by 2.5 dB in peak signal-to-noise ratio (PSNR) and surpassing existing Gaussian-based techniques by 3.5 dB in relighting tasks, all while maintaining a real-time rendering speed.

1. Introduction

Reconstructing 3D assets from images, or so-called *inverse rendering* (IR), is a longstanding task in computer graphics and vision. While recent techniques (e.g., Neural Radiance Fields (NeRF) [24] and 3D Gaussian Splatting (3DGS) [18]) have brought great opportunities to this task, it is still challenging due to its ill-posed nature and the complexity of both appearance and lighting.

Existing NeRF-based IR methods [2, 3, 10, 27, 29, 35, 37] have achieved impressive relighting quality, at the cost

of long training and rendering time. Recently, several methods [8, 13, 21, 26] have incorporated 3DGS into IR by decoupling radiance into material parameters for a Bidirectional Reflectance Distribution Function (BRDF) and environmental lighting to achieve relightability. Despite their fast training and rendering speeds, the quality of relighting often suffers from noticeable artifacts in both training and testing views (as shown in Fig. 8). The main reason for this issue is that each Gaussian has *constant normal material parameters* and leads to a uniform color across the entire Gaussian for given view and light directions. This constant radiance is against the fact that a Gaussian might cover different pixels that may have varying colors. We illustrate this observation in Fig. 2. Furthermore, these works model indirect illumination without considering physical constraints, leading to unnatural indirect lighting, which remains unchanged even under novel lighting conditions.

In this paper, we aim at a more capable representation for appearance modeling and a physically-based model for indirect lighting to enhance both novel view synthesis (NVS) and relighting quality. Inspired by the *flat shading* and *interpolated shading* in the domain of computer graphics, we propose a novel Gaussian representation that generalizes the per-Gaussian constant material to allow spatially-varying material. We refer to the former as the *Flat Gaussian* and the latter as the *Curved Gaussian* (CG). The key feature of our Curved Gaussian is that it allows for different normal and material properties at distinct representative locations, leading to a more powerful representation ability than the Flat Gaussian. Based on the Curved Gaussian, we design an inverse rendering framework called *CG-IR*, which consists of two key components. First, we introduce a novel rendering scheme for CGs, known as CG splatting, analogous to vertex/fragment shading in computer graphics. Second, we incorporate a physically-based indirect lighting model into our CG-IR framework by explicitly modeling light transport across different bounces, leading to a more reasonable decoupling of lighting and materials, as well as enabling indirect lighting with novel light sources. As a result, our CG-IR framework enhances both NVS and relighting quality, outperforming state-of-the-art NeRF-based methods by 2.5 dB in peak signal-to-noise ratio (PSNR) and surpassing Gaussian-based methods by 3.5 dB in the relighting task, maintaining a real-time rendering speed. To summarize, our contributions include:

- a novel representation—Curved Gaussian—capable of encoding spatially-varying material attributes on single primitive, enhancing representation ability,
- a new inverse rendering framework—CG-IR—built upon the Curved Gaussian representation, improving both NVS and relighting quality, and
- a physically-based indirect lighting model, which explicitly models light transport, resulting in more natural and realistic lighting.

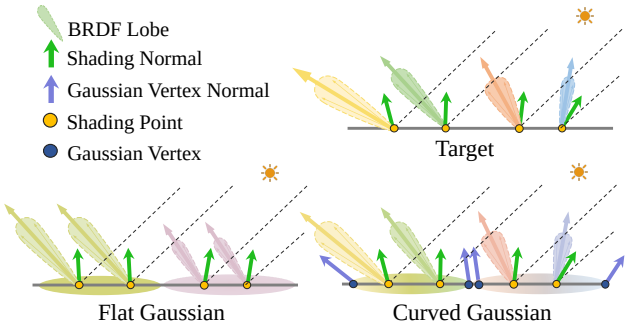


Figure 2. Illustration of two Flat Gaussians and Curved Gaussians fitting a distribution of BRDFs. As Curved Gaussians allow spatially-varying parameters, a single Gaussian can have different BRDF lobes at different places, leading to a more flexible representation compared to the Flat Gaussian.

2. Related Work

2.1. Neural Inverse rendering

Inverse rendering focuses on recovering material, geometry and lighting conditions from multi-view images. Several methods [2, 16, 30, 33, 34, 39] try to decompose lighting and material building upon the geometric formulation and volume rendering with NeRF [24]. Neural reflectance field [2] introduces BRDF and lighting in NeRF. PhySG [34] further refines the lighting model by using spherical Gaussians with incorporated SDFs for more precise geometry reconstruction. Next, several works have focused on modeling indirect illumination and visibility. MII [38] models indirect illumination by utilizing the neural radiance field. TensoIR [15] improves indirect lighting efficiency with a compact tri-plane representation for ray tracing. NeILF [30] and NeILF++ [33] represent incoming light as a neural incident light field. NeILF++ further combines VolSDF [31] with NeILF, incorporating inter-reflections for enhanced performance.

2.2. Gaussian splatting and relighting

Recently, 3DGS [19] introduced discretized explicit 3D Gaussian representations for scene modeling, leveraging differentiable rasterization for real-time rendering and achieving impressive results in the NVS task. However, the original 3DGS struggles to produce smooth and natural normals due to the lack of geometric constraints. Existing methods try to resolve this problem by utilizing 2D Gaussians as primitive [5, 11] or modeling a more accurate opacity field [32]. Better geometric representations facilitate applications like IR.

IR methods based on Gaussian splatting have emerged by incorporating BRDF into the attributes of Gaussians [9, 14, 22, 26]. Considering the illumination modeling, exist-

ing methods [9, 14, 22, 26] utilize one-step ray tracing or baked-based volumes to obtain visibility and leverage the unconstrained spherical harmonics (SH) to learn residual terms as the indirect illumination, which is challenging to achieve natural indirect lighting, due to the lack of physical constraints. Our approach not only extends the Gaussian representation but also models lighting physically, further improving the decoupling of material and lighting.

3. Preliminaries and motivation

Gaussian surfel splatting. Gaussian Surfel Splatting (GSS), also referred to as 2D Gaussian Splatting (2DGS), is a lower-dimensional form of 3DGS. 3DGS uses Gaussians to represent a scene, with the Gaussians defined as

$$\mathcal{G}(x) = e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}, \tag{1}$$

where μ and Σ denotes the mean and covariance matrix of the 3D Gaussian, representing the position and shape of the Gaussian points. Σ can be decomposed into the multiplication of a rotation matrix R and a scaling matrix S, as $\Sigma = RSS^TR^T$, while the Gaussian Surfels differ as the S is defined in 2-dimensional as,

$$S = \mathrm{Diag}(s_x, s_y, 0). \tag{2}$$

Similarly to 3DGS, the color C of each pixel is computed by alpha-blending all the ordered Gaussians overlapping the pixel:

$$T_i = \prod_{j=1}^{i-1}(1-\alpha_j), \tag{3}$$

$$C = \sum_{i=0}^{n} T_i c_i \alpha_i, \tag{4}$$

$$\alpha_i = \mathcal{G}^{'}(\mathbf{u}; \mathbf{u_i}, \Sigma_i^{'}) o_i. \tag{5}$$

where c_i is the color of each Gaussian, and α_i is a multiplication between opacity o_i and Gaussian Filter $\mathcal{G}^{'}$ [41] by the 2D covariance matrix in the screen space of the Gaussian surfels.

4. Method

In this section, we first introduce our new Curved Gaussian representation (Sec. 4.1). Next, we present our inverse rendering framework built on the CG representation (Sec. 4.2), followed by the physically-based illumination model (Sec. 4.3).

4.1. Curved Gaussian representation

For Gaussian-based inverse rendering approaches, we notice obvious artifacts in both training and testing views, as shown in Fig. 6. The reason behind this phenomenon is that, each Gaussian is tailored with a constant material parameters conducting an uniform color, which is not capable of capturing spatially-varying color across the area of the Gaussian. Therefore, we propose a novel Gaussian representation–*Curved Gaussian* (CG), which allows spatially-varying materials, to replace the original Gaussian (i.e., Flat Gaussian).

In our Curved Gaussian, we define several representative locations at each Gaussian, where these locations have different material parameters. Analogous to the concept of vertex for a triangle, we name a representative location as a *Gaussian vertex*. The Flat Gaussian is a special case in our definition, which only has one Gaussian vertex. We build the Gaussian vertex based on 2D Gaussian, parameterizing on the tangent space of each Gaussian. The tangent space is defined with rotation matrix R and then is stretched by the scaling matrix S. When querying attributes at any locations on the Gaussian, we interpolate the attribute values of Gaussian vertices to achieve spatially-varying properties.

By attaching attributes at Gaussian vertices, the capability of Gaussians can be enhanced to better fit a given distribution, as shown in Fig. 2.

4.2. CG-IR framework

Now, we introduce our inverse rendering framework–*CG-IR*, on top of the CG representation. The core of our framework is the rendering of Curved Gaussians and the physically-based illumination modeling, which will be presented in the next section.

We use Curved Gaussians with M Gaussian vertices as our Gaussian primitives, set as 4 in practice. Besides the parameters defined for Gaussians (covariance Σ_i, position μ, color c_i and opacity o_i), we introduce material parameters for each Gaussian vertex: albedo $\boldsymbol{a}_i$ and roughness r_i defined by the Disney Principled BRDF model [23]. We also set a normal offset ΔN_i at each Gaussian vertex. Here, the offset indicates the difference from the Gaussian normal N_i^g defined by Gaussian covariance. Now, Gaussian i has the following attributes: covariance Σ_i, position μ_i, radiance color c_i, opacity o_i, albedo $\boldsymbol{a}_i^{\{M\}}$, roughness $r_i^{\{M\}}$ and normal offset $\Delta N_i^{\{M\}}$.

Curved Gaussian splatting. Inspired by the rasterization of triangles, we propose a novel rendering scheme for CG, consisting of *Gaussian vertex shading* and *Gaussian fragment shading*. In the Gaussian vertex shading, the radiance of each Gaussian vertex is computed using the physically-based lighting (see Sec. 4.3), leading to M colors $\mathbf{c}^{\{M\}_i}$ for each Gaussian. Then, in the fragment shading, given a camera ray, the radiance of each Gaussian is computed by interpolating among M colors, followed by alpha blending, to obtain the radiance of each pixel.

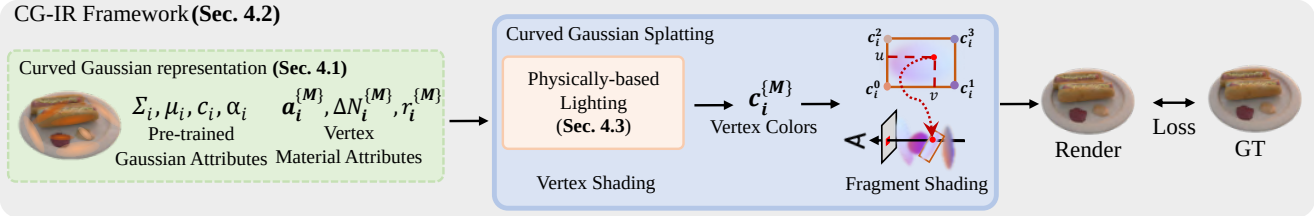


Figure 3. Overview of our framework. We propose a novel CG-IR framework. Within this framework, we introduce a Curved Gaussian representation capable of spatially variability with material attributes. We employ CG splatting, analogous to vertex and fragment shading in the traditional triangle rendering pipeline, to leverage the improved appearance capability of CGs. Additionally, we present a physics-based lighting model that enforces additional physical constraints to facilitate the decoupling of lighting and material properties.

Next, we provide details of Gaussian fragment shading. During rasterization, we compute the coordinates u, v in the tangent space of Gaussian i corresponding to pixel $\mathbf{u}$ in the screen space:

$$\mathbf{d}_i = \mathbf{J}_{pr}^{-1}\left(\mathbf{u} - \mathbf{u}_i\right), \tag{6}$$

$$(u, v) = \frac{\mathbf{d}_i}{S[:2] + \delta}, \tag{7}$$

where $\mathbf{d}_i$ is the displacement from the pixel center to the Gaussian center in the tangent space, $\mathbf{J}_{pr}^{-1}$ is the Jacobian of the inverse mapping from a pixel in the image space to the tangent space of each Gaussian and δ is an offset (set as 0.1).

In practice, we compute pixel's color by bilinear interpolation, followed by alpha blending:

$$\begin{aligned} \mathbf{c}_i &= \text{BilinearInterp}(\mathbf{c}_i^{\{M\}}, u, v), \\ C &= \sum_{i=0}^{n} \mathbf{c}_i \alpha_i T_i. \end{aligned} \tag{8}$$

Inverse rendering framework. With established Gaussian primitives and their rendering scheme, we build our inverse rendering framework, as shown in Fig. 3. Given multiple views, we utilize Gaussian Surfels [6] to optimize Gaussian attributes as the initialization. Then, we construct Curved Gaussians with attributes (Σ_i, μ_i, c_i, o_i) inherited from the pre-trained Gaussians and the material attributes of Gaussian vertices (i.e., $a_i^{\{M\}}$, $r_i^{\{M\}}$, $\Delta N_i^{\{M\}}$). Our framework optimizes the attributes of the Gaussian vertices. Using CG splatting, we render the images and compute the corresponding loss functions to guide the optimization.

4.3. Physically-based Illumination

A key of decomposing materials and lighting is the illumination model, including both direct lighting and indirect lighting, particularly for scenes with occlusions. Existing GS-based IR methods model the indirect illumination with trainable SHs which learn either the residual lighting [8, 22, 26] or the residual colors [13]. While these unconstrained manner improves NVS quality, the indirect lighting is unnatural, due to the lack of physical constraints. We address this issue by introducing physically-based illumination.

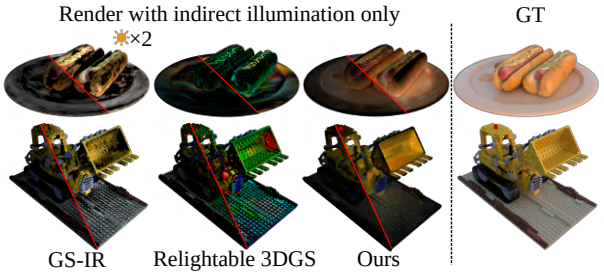


Figure 4. The results of rendering using only indirect lighting. We multiplied the brightness of the right half by two for better viewing. GS-IR fails to capture the effects of light reflections from objects as it only models brightness. Meanwhile, Relightable 3DGS lacks supervision for indirect lighting, leading to unnatural results. In contrast, our physically-based lighting model produces more natural rendering results.

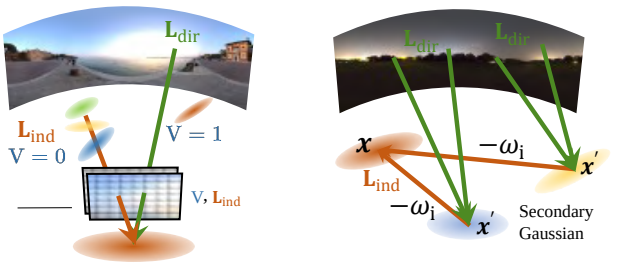
In our CG-IR, we compute the global illumination for each Gaussian vertex in the Gaussian vertex shading. Given a Gaussian vertex x, and view direction ω_o, its radiance $L_o(x, \omega_o)$ is defined by the rendering equation [17]:

$$L_o(x, \omega_o) = \int_{\Omega} f(x, \boldsymbol{a}, r, \omega_i, \omega_o) L_i(x, \omega_i)(\omega_i \cdot n) \mathrm{d}\omega_i, \tag{9}$$

where Ω is the upper hemisphere of the Gaussian defined by geometry normal N_i^g, f is the Disney Principled BRDF, $\boldsymbol{a}$ is albedo, r is roughness, ω_i is the incident direction and n is the shading normal of the shading point x.

We solve the above integral by uniformly sampling K directions within Ω, and then compute the incoming radiance $L_i(x, \omega_i)$ along each direction. Here, $L_i(x, \omega_i)$ is computed per Gaussian, instead of per Gaussian vertex, as the incoming radiance in a small region tends to be similar.

Per-Gaussian incoming radiance with raytracing. With the pre-trained Gaussian attributes in the initialization, we have the geometry and radiance field (although not



Physically-based Illumination

One-bounce Indirect Illumination When Relighting

Figure 5. Illustration of the physically-based illumination and our one-bounce relighting method when relighting.

relightable). We compute the incoming radiance L_i by ray tracing in the learned radiance field.

Specifically, we consider the Gaussians as ellipses, where the axis lengths are scaled by a factor of three. We then construct a bounding volume hierarchy (BVH) over ellipses to accelerate ray tracing. For each Gaussian, we uniformly sample K directions on its upper hemisphere. For each sampled direction, we emit a ray starting from the Gaussian center x with direction ω_i^k and query the Gaussians along the ray using the BVH. Then, we accumulate the transmittance T as Eqn. (3) and the radiance along each ray as L_i:

$$L_i(x,\omega_i) = L_i^{\text{dir}}(x,\omega_i)V(x,\omega_i) + L_i^{\text{ind}}(x,\omega_i), \quad (10)$$

$$V(x,\omega_i) = \begin{cases} 0, & T \le \varepsilon, \\ 1, & T > \varepsilon, \end{cases} \quad (11)$$

$$L_i^{\text{ind}}(x,\omega_i) = \sum_{l=0}^{n} c_l \alpha_l T_l, \quad (12)$$

where $L_{\text{dir}}(x,\omega_i)$ is the direct incoming radiance from the environment map, visibility to the environment map V is determined by T and ε is a threshold, set as 0.8 in practice. $L_{\text{ind}}(x,\omega_i)$ is the indirect radiance accumulated from Gaussian radiance field, c_l and α_l is the radiance field color and transparency of Gaussian l. Next, the radiance of Gaussian vertex is computed by Eqn. (9).

4.4. Relighting with CG-IR

The learned indirect illumination from previous works [8, 13, 22, 26] is specifically designed for the lighting conditions present during training. As a result, it cannot be used for relighting under new lighting conditions. In this section, we will show how to utilize our trained model for relighting in novel light environments.

Starting from Eqn. (10), Gaussians' radiance can not be used directly, as it's learned under the training light. Therefore, we replace L_i^{ind} in Eqn. (12) with one-bounce indirect illumination for relighting, as shown in Fig. 5.

For that, we query the Gaussian intersected with the ray at each direction, so called secondary Gaussian, and the incoming radiance $L_i^{\text{ind}}(x,\omega_i)$ at Gaussian vertex x is equivalent to the outgoing radiance $L_o(x',-\omega_i)$ of Gaussian k from $-\omega_i$:

$$\begin{aligned} L_i^{\text{ind}}(x,\omega_i) &= L_o(x',-\omega_i) \\ &= \sum_{i=0}^{K} 2\pi f'(x',\boldsymbol{a}',r',\omega_i^k,-\omega_i) L_{\text{dir}}(x',\omega_i^k) V(x',\omega_i^k), \end{aligned} \quad (13)$$

where f' represents the interpolated BRDF of vertices on Gaussian k. Finally, the radiance of Gaussian vertex is computed by Eqn. (10) and Eqn. (9). This allows us to obtain dynamic indirect illumination under the new light source. We show more implementation details in the supplementary material.

5. Implementation details

5.1. Radiance consistency loss

We observe that the indirect incoming radiance, obtained through ray tracing in the radiance field, can serve as a supervision for the outgoing radiance of secondary Gaussians along the sampled K directions. This provides additional guidance from extra viewpoints. However, performing one-bounce ray tracing for all Gaussians along the K directions is costly. We only sample one direction near the specular reflection for each Gaussian j. The supervision is applied as follows:

$$\mathcal{L}_{\text{rc}} = |L_i^{\text{ind}}(x_j,\omega_i^k), L_i^{\text{ind}'}(x_j,\omega_i^k)|, \quad (14)$$

where $L_i^{\text{ind}'}$ is the outgoing radiance of the secondary Gaussian on direction ω_i^k. For more details, please refer to the supplementary material.

5.2. Training details

We train our model using Adam optimizer [20] by the utilizing the losses as

$$\begin{aligned} \mathcal{L} =& \lambda_1 \mathcal{L}_1 + \lambda_{\text{ssim}} \mathcal{L}_{\text{ssim}} + \lambda_{\text{rc}} \mathcal{L}_{\text{rc}} \\ & \lambda_n \mathcal{L}_n + \lambda_{s,a} \mathcal{L}_{s,a} + \lambda_{s,r} \mathcal{L}_{s,r} + \lambda_{\text{reg},n} \mathcal{L}_{\text{reg},n}, \end{aligned} \quad (15)$$

where $\mathcal{L}_1$ and $\mathcal{L}_{\text{ssim}}$ is L_1 loss and SSIM loss between rendered image and ground truth, $\mathcal{L}_{\text{rc}}$ is the radiance consistency loss in Sec. 5.1, $\mathcal{L}_n$ is one minus cosine similarity between normal and pseudo normal, $\mathcal{L}_{s,a}$ and $\mathcal{L}_{s,r}$ are TV-loss on albedo and roughness, $\mathcal{L}_{\text{reg},n}$ is L_2 regular term of normal offsets ΔN^M. More details are in the supplementary.

6. Results

6.1. Evaluation setup

Dataset. To evaluate the capabilities on NVS and relighting of our method, we conduct experiments on both

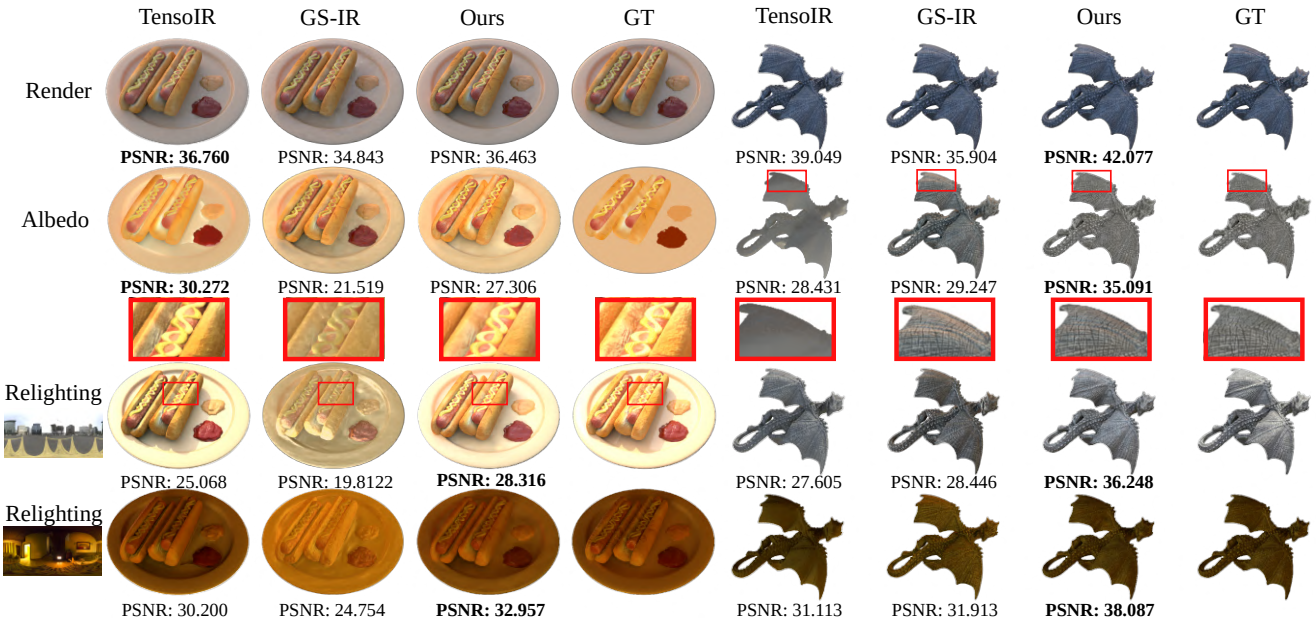


Figure 6. Qualitative comparison on TensoIR Synthetic and ADT datasets. Our method can provide more detailed relighting results than NeRF-based methods. And we alleviate the artifacts of Gaussian-based methods with better decoupling of material and illumination. More results are shown in the supplementary material.

Table 1. Comparison of relighting performance measured by PSNR↑, SSIM↑ and LPIPS↓ on the TensoIR Synthetic and Aria Digital Twin dataset. Numbers in red represent the best performance, while orange numbers denote the second best. Our method not only surpasses existing Gaussian-based approaches but also outperforms previous NeRF-based methods. Furthermore, we maintain the fast training speeds of Gaussian-based methods.

		NeRF-based		Gaussian-based			
		MII [39]	TensoIR [16]	GSshader [13]	GS-IR [22]	RelightGS [9]	Ours
		PSNR/ SSIM /LPIPS	PSNR/ SSIM /LPIPS	PSNR/ SSIM /LPIPS	PSNR/ SSIM /LPIPS	PSNR/ SSIM /LPIPS	PSNR/ SSIM /LPIPS
TensoIR	Armadillo	33.65 /0.9519/0.0643	34.41 /0.9753/0.0449	22.76 /0.9143/0.0716	29.39 /0.9204/0.0778	32.67 /0.9488/0.0697	35.01 /0.9629/0.0539
	Ficus	24.02 /0.9156/0.0758	24.30 /0.9466/0.0680	24.16 /0.9432/0.0468	25.45 /0.8934/0.0794	29.02 /0.9399/0.0430	31.13 /0.9543/0.0284
	Hotdog	28.91 /0.9156/0.0945	27.88 /0.9322/0.1160	18.02 /0.8782/0.1279	21.58 /0.8899/0.1244	22.19 /0.9072/0.1127	30.02 /0.9527/0.0649
	Lego	24.45 /0.8513/0.1462	27.54 /0.9249/0.0936	14.61 /0.7920/0.1294	22.33 /0.8389/0.1141	26.52 /0.8875/0.0988	28.19 /0.9141/0.0765
	Mean	27.76 /0.9086/0.0952	28.53 /0.9448/0.0806	19.89 /0.8819/0.0939	24.69 /0.8857/0.0989	27.60 /0.9209/0.0811	31.10 /0.9460/0.0558
ADT	Airplane	31.48 /0.9770/0.0252	32.63 /0.9810/0.0324	24.81 /0.9569/0.0333	28.63 /0.9558/0.0396	31.18 /0.9801/0.0283	35.98 /0.9845/0.0191
	Birdhouse	29.58 /0.9208/0.0837	31.59 /0.9551/0.0841	23.14 /0.9276/0.0531	25.11 /0.9006/0.0813	31.05 /0.9544/0.0551	31.85 /0.9592/0.0489
	Gargoyle	29.16 /0.9360/0.0567	30.42 /0.9609/0.0394	24.47 /0.9512/0.0356	27.78 /0.9496/0.0334	33.97 /0.9770/0.0203	36.25 /0.9837/0.0146
	Calculator	28.73 /0.9422/0.0546	27.66 /0.9549/0.0604	19.74 /0.8906/0.0818	26.26 /0.9323/0.0492	32.07 /0.9715/0.0285	33.56 /0.9795/0.0226
	Mean	29.74 /0.9440/0.0551	30.58 /0.9630/0.0541	23.04 /0.9316/0.0510	26.95 /0.9346/0.0509	32.84 /0.9707/0.3180	34.69 /0.9765/0.0275
	Training	6h	5h	0.5h	0.5h	1h	1h

synthetic and real-world datasets. For synthetic datasets, we benchmark on the commonly used TensoIR Synthetic datasets [16] and our collected relighting datasets of four real world scanned objects from Aria Digital Twin (ADT) [25]. This dataset will be released. And the real-world datasets include DTU [12] and MipNeRF360 [1]. Results on more datasets including NeRF Synthetic [24] and NeILF++ [33] are in the supplementary material.

Methods for comparison. We selected representative NeRF-based inverse rendering methods MII [39], TensoIR [16], as well as recent GS-based methods, such as GS-IR [22], GaussianShader [13] and Relightable 3DGS [9]. We trained each method using the official code and settings.

Metrics. We evaluate the quality of our comparison and ablation study results using PSNR, structural similarity in-

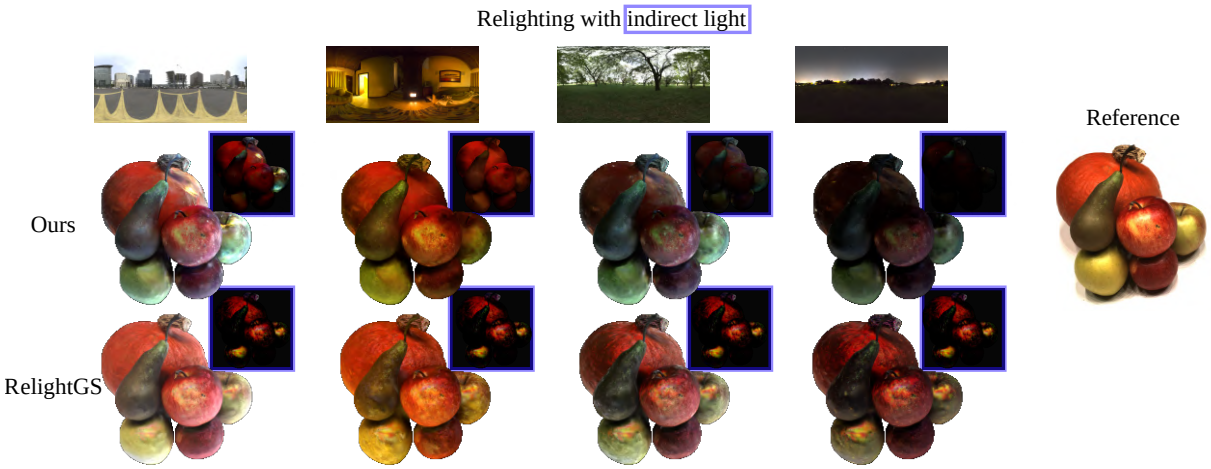


Figure 7. **Relighting results.** We perform relighting on real dataset DTU. The indirect shading results are in the blue borders. By recalculating indirect lighting for each new light source, our method avoids the unrealistic bright indirect illumination that Relightable 3DGS [8] exhibits under low-light conditions. More results are in the supplementary material.

Table 2. Comparison of NVS quality between our method and others on the TensoIR Synthetic and ADT dataset. Red numbers represent the best performance, and orange denotes the second best.

	TensoIR			ADT		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
MII	30.92	0.9371	0.0801	30.94	0.9568	0.0506
TensoIR	35.17	0.9764	0.0397	39.72	0.9916	0.0173
GS-IR	35.02	0.9637	0.0429	37.30	0.9821	0.0219
RelightGS	33.35	0.9657	0.0414	36.94	0.9879	0.0146
Ours	36.71	0.9758	0.0332	41.48	0.9923	0.0103

dex (SSIM) [28], and learned perceptual image patch similarity (LPIPS) [36]. To account for the ambiguity between albedo and lighting, we follow TensoIR by rescaling relighting images according to both the ground truth and predicted albedo. Additionally, we report the mean training time, both tested on the same one RTX 4090 setup.

6.2. Comparison with previous works

Relighting on synthetic data. In Tab. 1, we provide the numerical relighting results on the TensoIR Synthetic and ADT datasets. Our method achieves the SOTA relighting results in almost all scenes. In Fig. 6, we provide the visual comparison between our method and others. Our method produces smoother and more detailed relighting renderings, thanks to our Curved Gaussian representation and physically-based illumination. In contrast, GS-IR shows noticeable artifacts due to the constant color on a single Gaussian, and TensoIR produces renderings with missing details and unnatural lighting effects. Additional metrics on the datasets, along with more images, are provided in the supplementary materials.

Relighting on real data. For real scenes, we conducted experiments on several scenes selected from the object-level datasets DTU datasets and scene-level dataset Mip-NeRF360. Since ground truth is not available, we provide the reference training views along with the rendering results under novel lighting conditions. As shown in Fig. 7, the indirect lighting renderings produced by Relightable 3DGS under different environment maps remain the same, causing unnaturally bright renderings even under dark light sources. On the contrary, our method can dynamically adopt indirect lighting w.r.t. different environment maps due to our relighting strategy, which uses one-bounce ray tracing to model indirect illumination. Similarly, we compare our method with GS-IR in Fig. 9, showing more natural relighting results.

NVS. Besides relighting task, we validate our method on the NVS task, by performing a comparison with other IR methods [9, 16, 22, 38]. In Tab. 2, our method outperforms the others on the TensoIR Synthetic and ADT datasets. The visual comparison is shown in Fig. 8, showing that our method can produce cleaner results than GS-IR [22] and Reglitable 3DGS [9] with less artifacts, thanks to the spatially-varying material in a single Gaussian.

6.3. Ablation study

We perform ablation studies on the key components of our model on the TensoIR Synthetic dataset in terms of PSNR, SSIM and LPIPS. As shown in Tab. 3, we observe that as we incorporate our key components, the performance improves consistently.

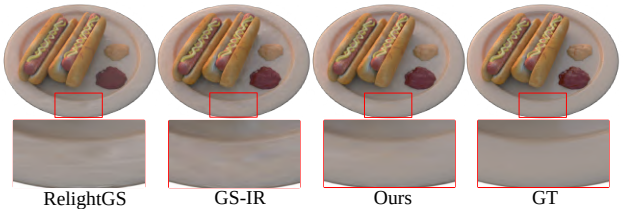


Figure 8. Comparison between our method and other GS-based IR methods on the NVS task. Our Curved Gaussian representation enhances the representation capacity, producing less artifacts.

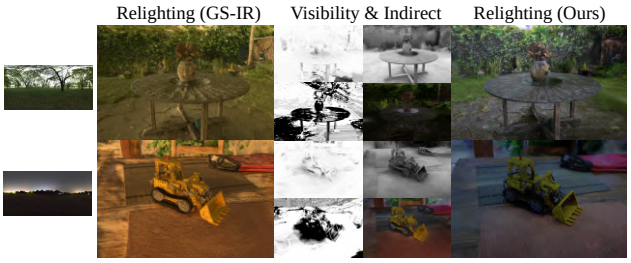


Figure 9. **Indirect Illumination on real dataset MipNeRF360.** Our physically-based indirect illumination modeling helps us achieve natural relighting results. More results are in the supplementary material.

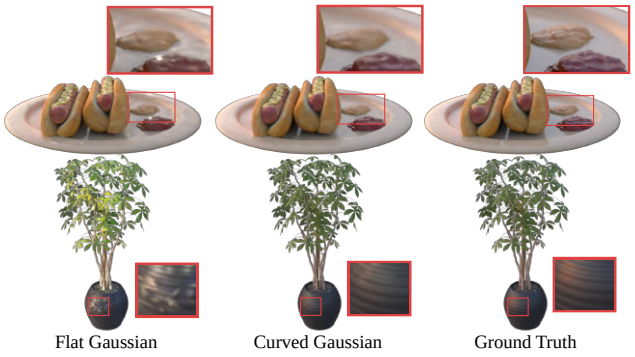


Figure 10. Ablation study on Curved Gaussian representation. The Curved Gaussian yields smoother rendering results, significantly reducing artifacts.

Curved Gaussian representation. We evaluate the impact of our Curved Gaussian representation in Tab. 3 and Fig. 10. As shown in Fig 10, our Curved Gaussian shows a smoother surface, and fewer artifacts compared to the Flat Gaussian.

Indirect illumination modeling. Besides the metrics improvement shown in Tab. 3, we further conducted a more in-depth ablation study on components (visibility, indirect illumination) of our physically-based indirect illumination, as shown in Tab. 4. As the components were gradually added, both the relighting and NVS quality progressively improved. The specific impact of each component on the training process is explained in the supplementary.

Table 3. Ablation study of our key components on TensoIR Synthetic dataset. "CG" means the Curved Gaussian representation, "PBI" means the physically-based illumination. Numbers in red represent the best performance, while orange numbers denote the second best.

Component		Relighting			NVS		
CG	PBI	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✗	✗	28.614	0.904	0.093	34.640	0.941	0.047
✓	✗	29.447	0.937	0.074	35.794	0.961	0.041
✓	✓	31.087	0.946	0.055	36.709	0.975	0.033

Table 4. Ablation study of several components in our physically-based lighting model on the TensoIR Synthetic dataset. "Vis." means the visibility to environment light, "Ind." means the radiance field indirect illumination term. Numbers in red represent the best performance, while orange numbers denote the second best.

Component		Relighting			NVS		
Vis.	Ind.	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✗	✗	29.447	0.937	0.074	35.794	0.961	0.041
✓	✗	30.253	0.941	0.068	36.059	0.963	0.037
✓	✓	31.087	0.946	0.055	36.709	0.975	0.033

6.4. Discussion and limitations

We have identified several limitations of our method. We do not introduce geometric prior, leading to unsatisfied recovery for highly specular objects. This issue can be alleviated by introducing SDF like GS-ROR [40], which we leave for future work. Additionally, our model brings more GPU memory usage (50% $\sim$ 80%) than 3DGS/2DGS and 10% $\sim$ 20% more than Relightable 3DGS.

7. Conclusion

In this paper, we have presented a novel framework for inverse rendering. At the core of our framework is a curved Gaussian representation and a curve Gaussian rendering scheme, which allows spatially-varying material and normal parameters, showing more powerful representation capability than typical Gaussian primitives. Besides, we present a physically-based indirect illumination modeling method, which brings physical constraints for training and enables natural indirect light for relighting. Consequently, our method outperforms existing NeRF-based and Gaussian-based inverse rendering methods without losing efficiency. In future work, improving the geometric accuracy is a potential direction.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5470–5479, New York, NY, USA, 2022. IEEE. 6, 2

[2] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition, 2020. 1, 2

[3] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan Barron, and Hendrik PA Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *NeurIPS*, pages 10691–10704, Red Hook, NY, USA, 2021. Curran Associates, Inc. 1

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, page 333–350, Berlin, Heidelberg, 2022. Springer. 2

[5] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 2

[6] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *SIGGRAPH*, New York, NY, USA, 2024. Association for Computing Machinery. 4, 1, 2

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, New York, NY, USA, 2022. IEEE. 2

[8] 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. *arXiv:2311.16043*, 2023. 2, 4, 5, 7

[9] Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing, 2024. 2, 3, 6, 7, 1

[10] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising. *Advances in Neural Information Processing Systems*, 35:22856–22869, 2022. 1

[11] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, New York, NY, USA, 2024. Association for Computing Machinery. 2

[12] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 6, 2

[13] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5322–5332, 2024. 2, 4, 5, 6, 1

[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. In *CVPR*, pages 5322–5332, New York, NY, USA, 2024. IEEE. 2, 3

[15] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensoir: Tensorial inverse rendering. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2023. 2

[16] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensoir: Tensorial inverse rendering. In *CVPR*, pages 165–174, New York, NY, USA, 2023. IEEE. 2, 6, 7, 3

[17] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986. 4

[18] 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

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):139–1, 2023. 2, 1

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[21] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21644–21653, 2024. 2

[22] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *CVPR*, pages 21644–21653, New York, NY, USA, 2024. IEEE. 2, 3, 4, 5, 6, 7, 1

[23] Stephen McAuley, Stephen Hill, Adam Martinez, Ryusuke Villemin, Matt Pettineo, Dimitar Lazarov, David Neubelt, Brian Karis, Christophe Hery, Naty Hoffman, et al. Physically based shading in theory and practice. In *ACM SIGGRAPH 2013 Courses*, pages 1–8. 2013. 3

[24] 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*, pages 405–421, Berlin, Heidelberg, 2020. Springer. 1, 2, 6, 3

[25] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception, 2023. 6

[26] 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, 2023. 2, 3, 4, 5

[27] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting

and view synthesis. In *CVPR*, pages 7495–7504, New York, NY, USA, 2021. IEEE. 1

[28] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 7

[29] Ziyi Yang, Yanzhen Chen, Xinyu Gao, Yazhen Yuan, Yu Wu, Xiaowei Zhou, and Xiaogang Jin. Sire-ir: Inverse rendering for brdf reconstruction with shadow and illumination removal in high-illuminance scenes, 2023. 1

[30] Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Neilf: Neural incident light field for physically-based material estimation. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[31] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, pages 4805–4815, Red Hook, NY, USA, 2021. Curran Associates, Inc. 2

[32] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024. 2

[33] 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 *ICCV*, pages 3601–3610, New York, NY, USA, 2023. IEEE. 2, 6

[34] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *CVPR*, pages 5453–5462, New York, NY, USA, 2021. IEEE. 2

[35] Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. Iron: Inverse rendering by optimizing neural sdfs and materials from photometric images. In *CVPR*, pages 5565–5574, New York, NY, USA, 2022. IEEE. 1

[36] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, New York, NY, USA, 2018. IEEE. 7

[37] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: neural factorization of shape and reflectance under an unknown illumination. *ACM TOG*, 40(6), 2021. 1, 4, 5

[38] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18622–18631, 2022. 2, 7

[39] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *CVPR*, pages 18643–18652, New York, NY, USA, 2022. IEEE. 2, 6

[40] Zuo-Liang Zhu, Beibei Wang, and Jian Yang. Gs-ror: 3d gaussian splatting for reflective object relighting via sdf priors. *arXiv preprint arXiv:2406.18544*, 2024. 8

[41] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 3

CG-IR: Curved Gaussian Splatting for Inverse Rendering

Supplementary Material

We propose CG-IR, a novel inverse rendering framework based on our proposed Curved Gaussian representation with spatially-varying material attributes. Besides we introduce our physically-based illumination to better decouple material properties and illumination. In this supplementary material, we provide implementation details Sec. 8, along with additional results Sec. 9 and ablation study Sec. 10.

8. Implementation details

Curved Gaussian representation. As described in Sec. 4.1. In our Curved Gaussian, we define several Gaussian vertices with different material attributes on a single Gaussian primitive. The Gaussian vertices are parameterized on the tangent space of each Gaussian defined by the rotation matrix R and the scaling matrix S. In implementation, we create a square on each Gaussian surfel, where the length of each side of the square is determined by twice the scaling S along the two axes.

Ray tracing. We perform ray tracing on Gaussian surfels in Sec. 4.3 to obtain the incoming radiance of each Gaussian. When constructing the BVH, we treat each Gaussian as an elliptical disk, where the lengths of the two axes are set to three times the scaling factor, and the thickness of each disk is defined as 1×10^{-12}. For each Gaussian, we emit rays along the K sampled directions as well as along the upper hemisphere to perform ray intersections. When a ray hits a Gaussian, the new ray's starting point is the hit point, plus an offset of ϵ (set as 0.05) times the ray direction, and the ray tracing continues from there. The radiance and transmittance are accumulated as Eqn. (12) and Eqn. (3). The ray tracing results are stored in the micro-buffers of each Gaussian for later direct queries, eliminating the need to re-trace the rays. Besides, during ray tracing, we also record the index I of the first Gaussian hit and its coordinates in the tangent space U, which are stored in the Gaussian's buffer as well.

One-bounce indirect illumination. We replace the indirect illumination from the radiance colors of Gaussians with one-bounce indirect illumination in Sec. 4.4. Thanks to the micro-buffers stored in ray-tracing, we can utilize the index buffers I to query the bounce between Gaussians quickly with the tangent space coordinates U to determine the weights of interpolation. Then we can compute the one-bounce indirect illumination in Eqn. (13) with a fast speed.

Radiance consistency loss. We leverage one-bounce indirect illumination as a supervision when training by sampling the specular direction as

$$k_j = \text{argmax}(< \omega_o^k, 2\boldsymbol{N}_o^g - \omega_i > \ \& \ \ V_j^k = 0) \tag{16}$$

where ω_o^k represents the view direction on Gaussian j. We only sample on directions that are not visible to direct light, ensuring the presence of indirect lighting from bounces between Gaussians. Thus, We can compute the radiance consistency loss as Eqn. (14). As described in Sec. 8, the index micro buffers also reduce the computation time this loss less than 1 ms.

Loss details. We train Gaussian vertex attributes using loss terms in Eqn. (15). The $\mathcal{L}_1$ and $\mathcal{L}_{\text{ssim}}$ reprensent L_1 loss and SSIM loss between rendered image and ground truth, which are commonly used rendering loss by previous methods [9, 19, 22]. $\mathcal{L}_{\text{rc}}$ is the radiance consistency loss defined in Eqn. (14). $\mathcal{L}_{s,a}$ is TV-loss on albedo for smoothness, defined as

$$\begin{aligned} \Delta_{ij}^{\hat{\alpha}} =& \exp\left(-|I_{i,j} - I_{i-1,j}|\right)\left(\hat{\alpha}_{i,j} - \hat{\alpha}_{i-1,j}\right)^2 + \\ & \exp\left(-|I_{i,j} - I_{i,j-1}|\right)\left(\hat{\alpha}_{i,j} - \hat{\alpha}_{i,j-1}\right)^2, \\ \mathcal{L}_{s,a} =& \frac{1}{|\hat{\alpha}|}\sum_{i,j}\Delta_{ij}^{\hat{\alpha}}, \end{aligned} \tag{17}$$

where $\hat{\alpha}$ is the albedo map obtained by the CG splatting in Sec. 4.2. $\mathcal{L}_{s,r}$ is the TV-loss on roughness similar to $\mathcal{L}_{s,a}$. $\mathcal{L}_n$ is the normal consistency loss in Gaussian surfels [6] by

$$\mathcal{L}_n = (1 - \hat{n}^\top \hat{n}_{\hat{D}}) \tag{18}$$

where $\hat{n}$ is normal map and $\hat{n}_{\hat{D}}$ is the pseudo normal map from the depth map. $\mathcal{L}_{\text{reg},n}$ is L_2 regular term of normal offsets from Gaussian Shader [13] as

$$\mathcal{L}_{\text{reg},n} = ||\Delta N^{\{M\}}||^2 \tag{19}$$

The loss weights $\{\lambda_1, \lambda_{\text{ssim}}, \lambda_{\text{rc}}, \lambda_n, \lambda_{s,a}, \lambda_{s,r}, \lambda_{reg,n}\}$ are set as $\{0.9, 0.1, 0.05, 0.02, 0.1, 0.05, 0.01\}$.

9. More results

Results on TensoIR Synthetic dataset. We show more inverse rendering results on Figs. 12 to 15. The metrics on albedo and normal are shown in Tab. 5. Relightable 3DGS [9] conducts over-smooth normal and albedo. GaussianShader [13] produces unnatural relighting results. Due to the residual color terms and the approximation of PBR.

Table 5. Comparison of albedo and normal on TensoIR Synthetic and ADT datasets. Numbers in red represent the best performance, while orange numbers denote the second best.

	Method	Albedo PSNR↑ / SSIM↑ / LPIPS↓	Normal MAE↓
TensoIR	MII	27.293 / 0.933 / 0.101	5.076
	TensoIR	29.275 / 0.950 / 0.087	4.098
	Gsshader	25.026 / 0.923 / 0.087	5.757
	GS-IR	30.286 / 0.941 / 0.084	5.341
	RelightGS	28.537 / 0.922 / 0.087	5.064
	Ours	30.341 / 0.951 / 0.074	4.358
ADT	MII	29.150 / 0.952 / 0.068	3.027
	TensoIR	29.295 / 0.954 / 0.056	2.688
	Gsshader	30.432 / 0.960 / 0.036	1.995
	GS-IR	32.711 / 0.968 / 0.037	2.665
	RelightGS	21.047 / 0.911 / 0.039	2.179
	Ours	33.630 / 0.980 / 0.023	1.703

GS-IR [22] produces coarse normals and albedo with baked-in lighting effects. TensoIR [16] lacks the details in rendering, e.g. the texture on the bread in the hotdog scene. Our method leverages Curved Gaussians and physically-based illumination to enhance representational capacity and lighting decoupling, achieving high-quality results on both relighting and NVS. We also provide detailed per-scene results in Tab. 9.

Results on ADT dataset. We show more inverse rendering results on Figs. 16 to 19. The metrics on albedo and normal are shown in Tab. 5. We achieve excellent inverse rendering and relighting results thanks to our CG-IR framework in ADT dataset. We also provide detailed per-scene results in Tab. 10.

Results on DTU dataset. In Fig. 20, we demonstrate the inverse rendering results of our method on the real-world DTU dataset [12]. Utilizing our CG-IR framework, we recover the material properties and achieve high-quality relighting. Besides, our approach produces natural indirect lighting, thanks to our physically-based illumination model.

Results on NeILF++ dataset. To verify its robustness in relighting, we further evaluate our method on the real-world dataset NeILF++ [33], scomparing it against GS-IR [21] and Relightable 3DGS [9], as shown in Fig. 21. The results demonstrate the robustness of our CG-IR framework, achieving high-quality relighting even on the relatively sparse-view real-world dataset.

Results on Mip-NeRF360 dataset. Fig. 22 showcases the inverse rendering and relighting results of our method on MipNeRF360 datasets [1]. Our CG-IR framework achieves high-quality reconstruction with the Curved Gaussian and physically-based illumination, demonstrating robust performance on scene-level real-world datasets.

Table 6. Comparison of NVS quality between our method and others on NeRF Synthetic dataset. Red numbers represent the best performance, and orange denotes the second best.

Method	Relightable	PSNR↑	SSIM↑	LPIPS↓
NeRF [24]	✗	31.012	0.947	0.056
Plenoxels [7]	✗	31.714	0.958	0.053
TensoRF [4]	✗	33.140	0.963	0.042
3DGS [19]	✗	33.883	0.971	0.031
GaussianSurfels [6]	✗	33.053	0.961	0.036
TensoIR [16]	✓	29.537	0.943	0.067
GaussianShader [13]	✓	33.367	0.960	0.042
RelightGS [9]	✓	28.238	0.938	0.056
GS-IR [22]	✓	30.133	0.937	0.059
Ours	✓	30.338	0.946	0.051

Table 7. Ablation study of the loss terms. Numbers in red represent the best performance, while orange numbers denote the second best.

Component			Relighting			NVS		
$\mathcal{L}_N$	$\mathcal{L}_s$	$\mathcal{L}_{rc}$	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✗	✓	✓	30.002	0.939	0.062	36.379	0.972	0.034
✓	✗	✓	30.753	0.941	0.054	36.722	0.975	0.035
✓	✓	✗	30.662	0.943	0.061	36.444	0.974	0.035
✓	✓	✓	31.087	0.946	0.055	36.709	0.975	0.033

Results on NeRF Synthetic dataset. We evaluate the NVS performance of our method on the NeRF Synthetic dataset [24] and compare it with both relightable and non-relightable approaches in Tab. 6. Among relightable methods, our approach achieves near SOTA NVS quality. GaussianShader builds upon the radiance SH representation in 3DGS by incorporating BRDF, rather than replacing radiance SH with BRDF as other methods do. This enables it to achieve the NVS quality comparable to the original 3DGS at the expense of relighting performance. In contrast, our Curved Gaussian representation and physically-based illumination enable competitive NVS quality while preserving relighting fidelity.

10. More ablation study

Loss. We perform ablation experiments on the loss functions to analyze their impact. The loss terms outlined in Sec. 8 are divided into three categories: (1) the normal loss $\mathcal{L}_N$, which includes $\mathcal{L}_n$; (2) the smoothness loss $\mathcal{L}_s$, comprising $\mathcal{L}_{s,a}$ and $\mathcal{L}_{s,r}$; and (3) our proposed radiance consistency loss $\mathcal{L}_{rc}$. The metrics are shown in Tab. 7. Normals require constraints to prevent overfitting due to their inherent ambiguity in appearance representation. Smooth

Table 8. Ablation on the sample counts. We present the relighting quality and corresponding cost at different sample counts K. Numbers in red represent the best performance, while orange numbers denote the second best. In practice, we set $K = 64$.

Sample count	Relighting			Cost	
	PSNR↑	SSIM↑	LPIPS↓	Memory↓	Rendering Time↓
$K = 16$	34.114	0.9587	0.05627	11.1GB	10ms
$K = 32$	34.842	0..96112	0.05542	12.1GB	11ms
$K = 64$	35.010	0..96289	0.05401	14.3GB	13ms
$K = 128$	35.009	0.96311	0.05392	20.4GB	21ms

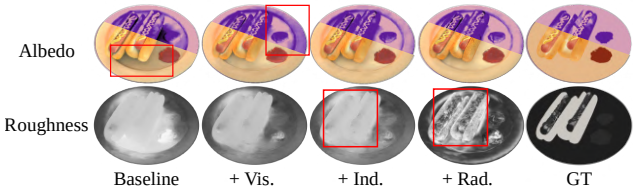


Figure 11. Ablation of indirect illumination components and radiance consistency loss. "Vis." means the visibility, "Ind." means the indirect illumination and "Rad." means the radiance consistency loss. The right part of albedo maps are processed with higher contrast for better observation. The GT roughness is from the "hotdog" blender scene in NeRF Synthetic dataset [24] rather than the TensoIR dataset. [16]

loss terms on material parameters lead to cleaner rendering results than those without such regularization, which is beneficial for relighting. The radiance consistency loss leverages the pre-trained radiance field to provide supervision from additional viewpoints, improving the quality of both relighting and NVS.

Ray sample counts and cost. We conduct additional experiments on the counts of the sampled directions for per Gaussian. We evaluate the quality, memory cost and rendering time on "armadillo" from TensoIR Synthetic as shown in Tab 8. Under the observation that the quality reaches a plateau at $K = 64$, while maintaining real-time rendering speed and acceptable memory usage, we finally select $K = 64$ for the balance of the quality and the cost.

Indirect illumination. Fig. 11 presents the ablation results for albedo and roughness maps. By modeling visibility, we alleviate the issue of shadows being baked into the albedo in baseline methods. Indirect illumination modeling further helps decoupling of material and lighting, preventing discrepancies in albedo caused by differing lighting conditions from the left to right. Moreover, indirect illumination serves as the foundation of our proposed radiance consistency loss $\mathcal{L}_{rad}$, which ensures roughness aligns more closely with the ground truth. This improvement is achieved through the additional viewpoint guidance provided by $\mathcal{L}_{rad}$.

Table 9. Per-scene results of normal, albedo and NVS on TensoIR Synthetic dataset. For albedo results, we follow NeRFactor [37] by scaling each RGB channel by a global scalar.

Scene	Method	Normal	Albedo			Novel View Synthesis		
		MAE↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Armadillo	InvRender	1.732	35.573	0.959	0.076	36.681	0.971	0.056
	TensoIR	1.960	34.360	0.989	0.059	39.070	0.986	0.039
	GSshader	2.107	31.092	0.938	0.053	42.445	0.989	0.024
	GS-IR	3.105	38.572	0.986	0.051	38.530	0.972	0.041
	RelightGS	2.224	34.435	0.933	0.067	39.440	0.980	0.042
	Ours	1.974	36.851	0.973	0.047	41.057	0.983	0.031
Ficus	InvRender	4.884	25.335	0.942	0.072	25.498	0.939	0.062
	TensoIR	4.400	27.130	0.964	0.044	29.770	0.973	0.041
	GSshader	4.513	28.239	0.966	0.028	35.256	0.990	0.012
	GS-IR	5.104	30.867	0.948	0.053	33.258	0.960	0.039
	RelightGS	4.991	28.597	0.912	0.057	32.405	0.974	0.028
	Ours	3.408	31.580	0.972	0.032	34.899	0.978	0.025
Hotdog	InvRender	3.708	27.028	0.950	0.094	32.219	0.952	0.070
	TensoIR	4.050	30.370	0.947	0.099	36.780	0.976	0.046
	GSshader	8.315	18.149	0.909	0.127	36.897	0.980	0.029
	GS-IR	4.774	26.745	0.941	0.088	34.843	0.969	0.051
	RelightGS	5.399	25.277	0.939	0.087	30.371	0.943	0.045
	Ours	4.016	27.252	0.952	0.078	36.329	0.977	0.034
Lego	InvRender	9.980	21.435	0.882	0.160	28.277	0.887	0.133
	TensoIR	5.980	25.240	0.900	0.145	35.040	0.970	0.033
	GSshader	8.094	22.625	0.877	0.140	35.403	0.976	0.024
	GS-IR	8.380	24.958	0.889	0.143	33.455	0.954	0.042
	RelightGS	7.643	25.838	0.902	0.135	30.371	0.943	0.045
	Ours	8.032	25.681	0.901	0.139	34.551	0.964	0.041

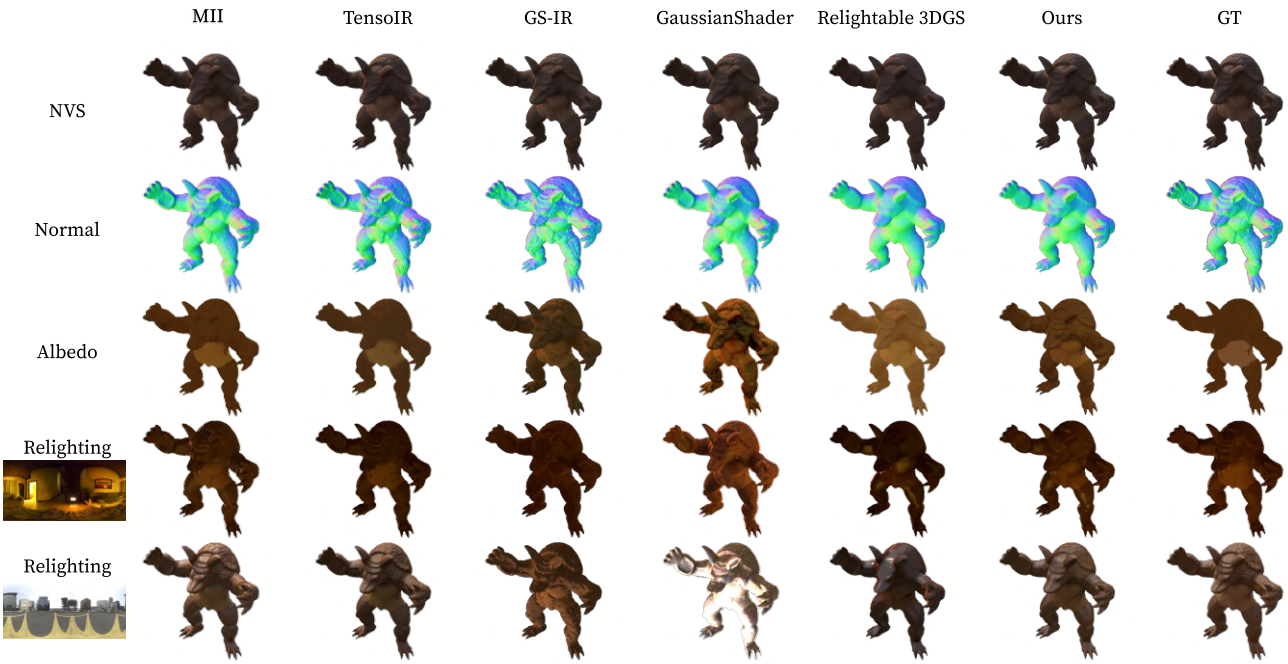


Figure 12. Qualitative comparison of NVS, normal, albedo and relighting on **armadillo** of TensoIR Synthetic dataset.

Table 10. Per-scene results of normal, albedo and NVS on ADT dataset. For albedo results, we follow NeRFactor [37] by scaling each RGB channel by a global scalar.

Scene	Method	Normal	Albedo			Novel View Synthesis		
		MAE ↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Airplane	InvRender	1.688	30.240	0.978	0.037	32.794	0.985	0.022
	TensoIR	1.320	32.400	0.983	0.022	40.370	0.995	0.011
	GSshader	1.207	33.233	0.974	0.024	44.640	0.997	0.004
	GS-IR	1.584	35.449	0.978	0.035	38.755	0.985	0.020
	RelightGS	1.298	35.375	0.973	0.034	37.982	0.991	0.01
	Ours	0.876	36.172	0.987	0.017	42.568	0.994	0.007
Birdhouse	InvRender	3.912	27.770	0.948	0.107	31.237	0.943	0.076
	TensoIR	2.960	29.350	0.961	0.084	39.350	0.986	0.031
	GSshader	3.148	25.984	0.929	0.061	42.167	0.990	0.016
	GS-IR	4.811	28.466	0.944	0.057	37.057	0.977	0.033
	RelightGS	3.083	25.245	0.939	0.055	36.935	0.982	0.027
	Ours	2.911	29.674	0.963	0.04	40.395	0.987	0.019
Gargoyle	InvRender	2.982	29.064	0.924	0.066	29.874	0.945	0.054
	TensoIR	3.310	28.430	0.923	0.067	39.050	0.993	0.010
	GSshader	1.616	30.846	0.972	0.024	42.497	0.996	0.004
	GS-IR	1.711	31.955	0.973	0.022	35.904	0.984	0.013
	RelightGS	2.253	31.424	0.931	0.025	38.910	0.989	0.007
	Ours	1.581	35.09	0.989	0.012	42.079	0.995	0.005
Calculator	InvRender	3.526	29.526	0.956	0.061	29.854	0.954	0.050
	TensoIR	3.160	27.000	0.949	0.051	40.100	0.993	0.016
	GSshader	2.007	31.665	0.965	0.036	43.857	0.996	0.005
	GS-IR	2.553	34.973	0.976	0.032	37.470	0.983	0.022
	RelightGS	2.081	27.216	0.948	0.042	33.915	0.986	0.013
	Ours	1.445	33.582	0.980	0.021	40.881	0.993	0.009

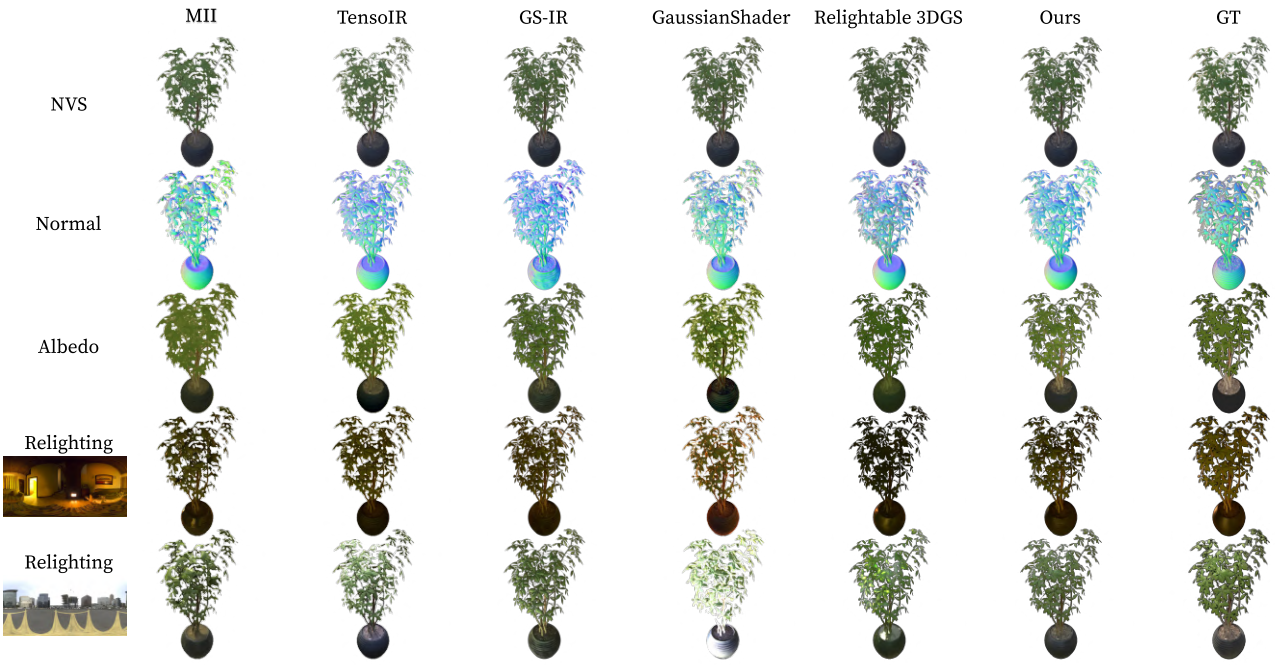


Figure 13. Qualitative comparison of NVS, normal, albedo and relighting on **ficus** of TensoIR Synthetic dataset.

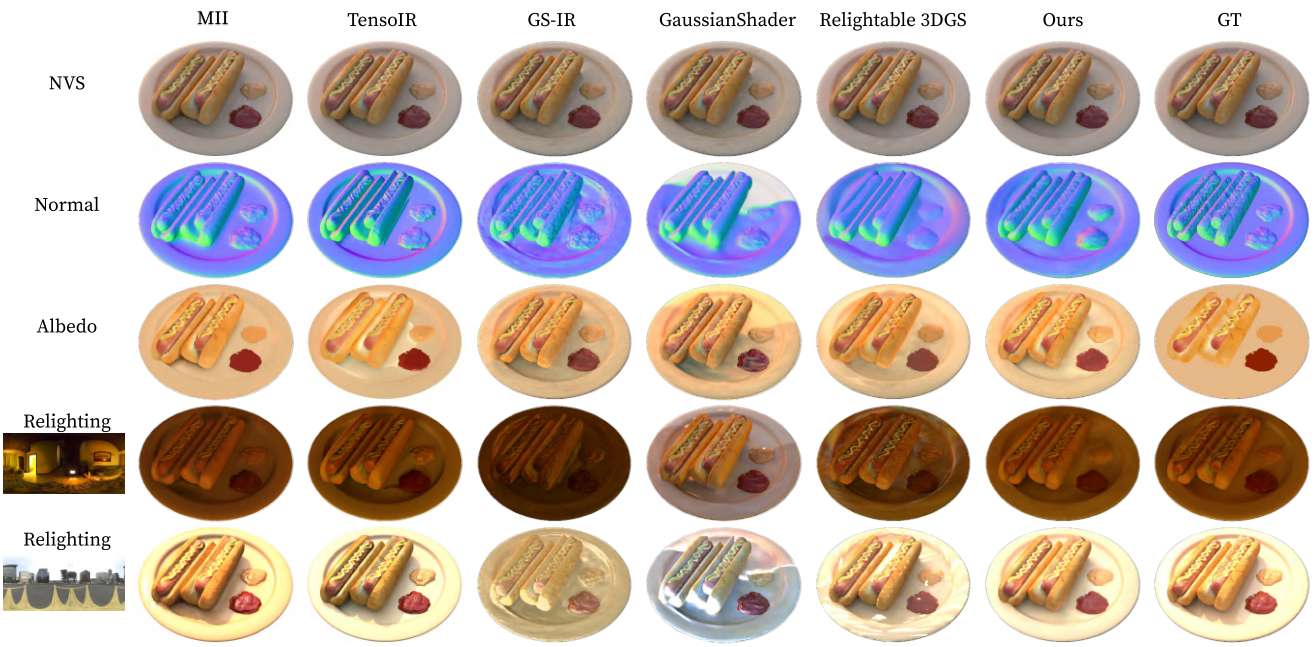


Figure 14. Qualitative comparison of NVS, normal, albedo and relighting on **hotdog** of TensoIR Synthetic datasets.

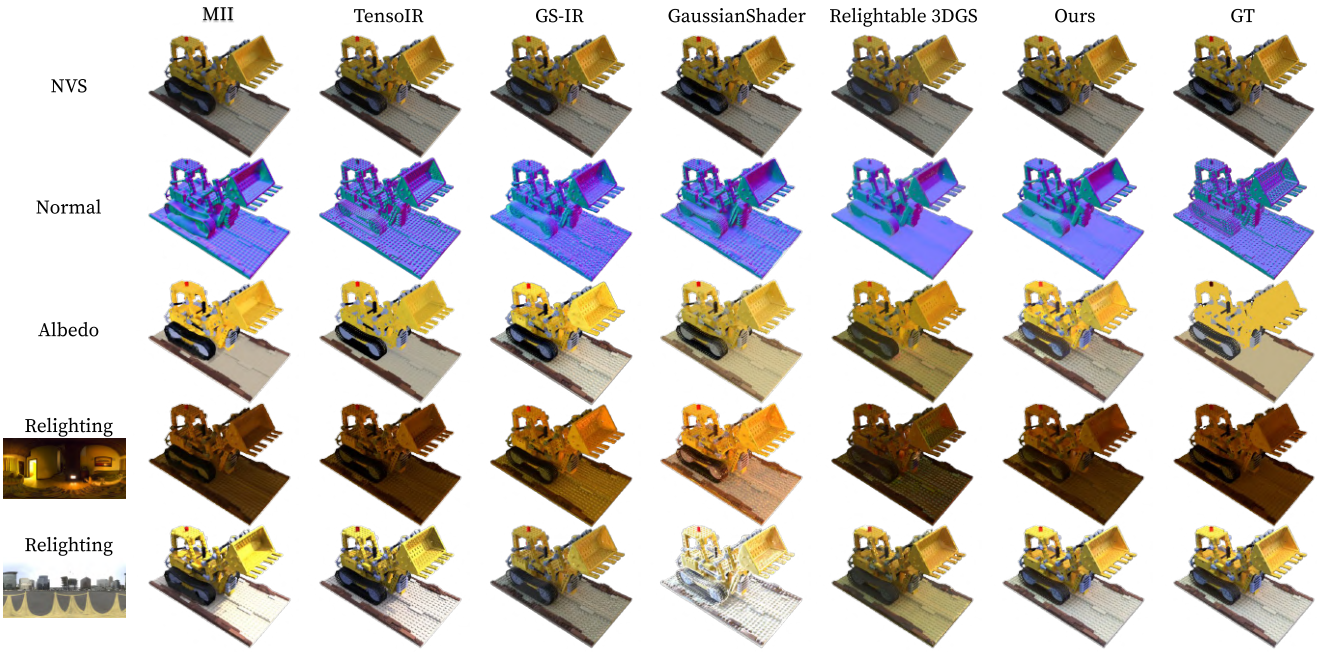


Figure 15. Qualitative comparison of NVS, normal, albedo and relighting on **lego** of TensoIR Synthetic dataset.

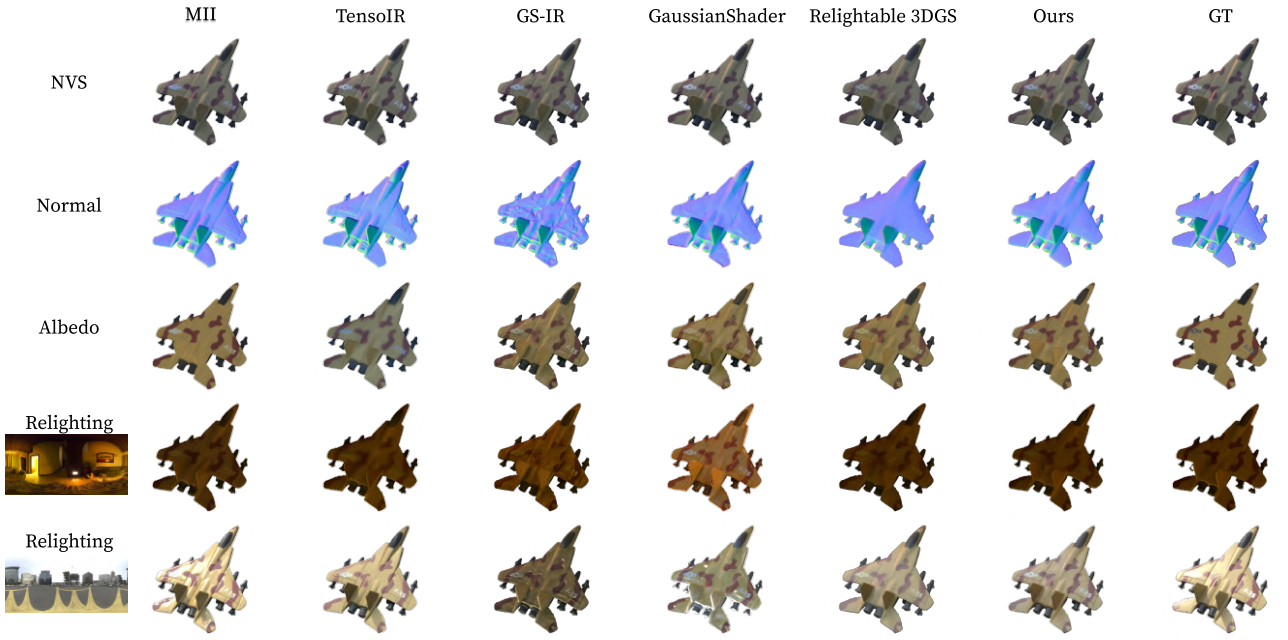


Figure 16. Qualitative comparison of NVS, normal, albedo and relighting on **airsplane** of ADT dataset.

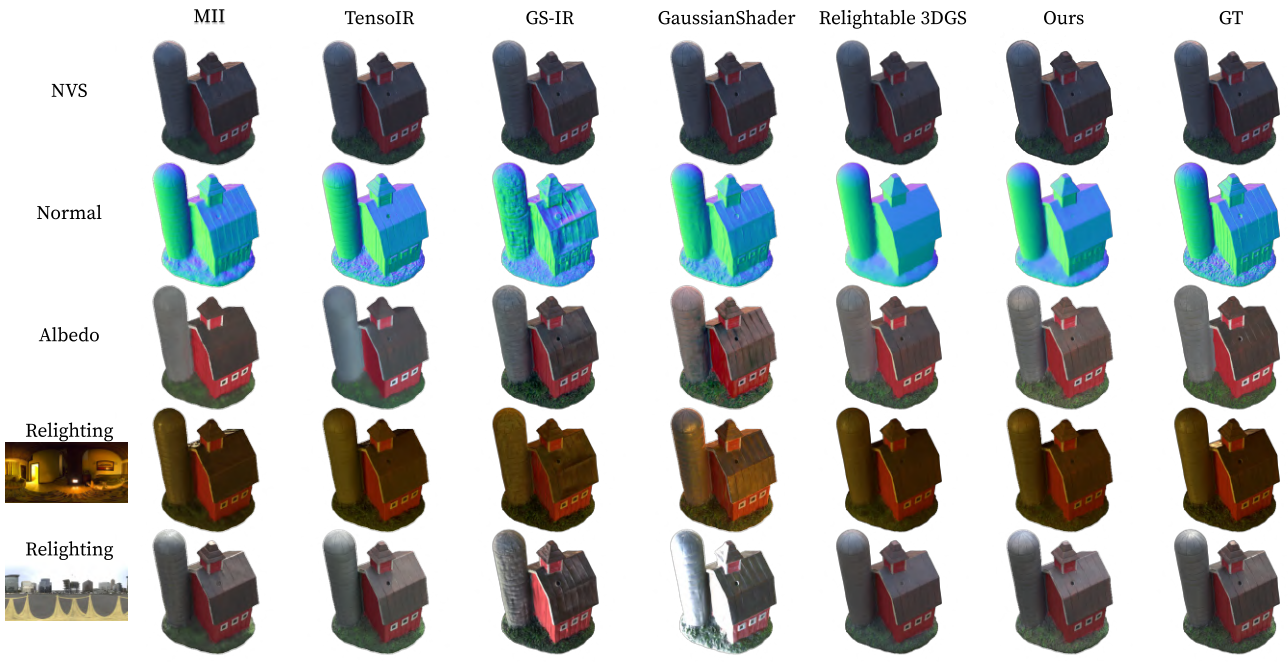


Figure 17. Qualitative comparison of NVS, normal, albedo and relighting on **birdhouse** of ADT dataset.

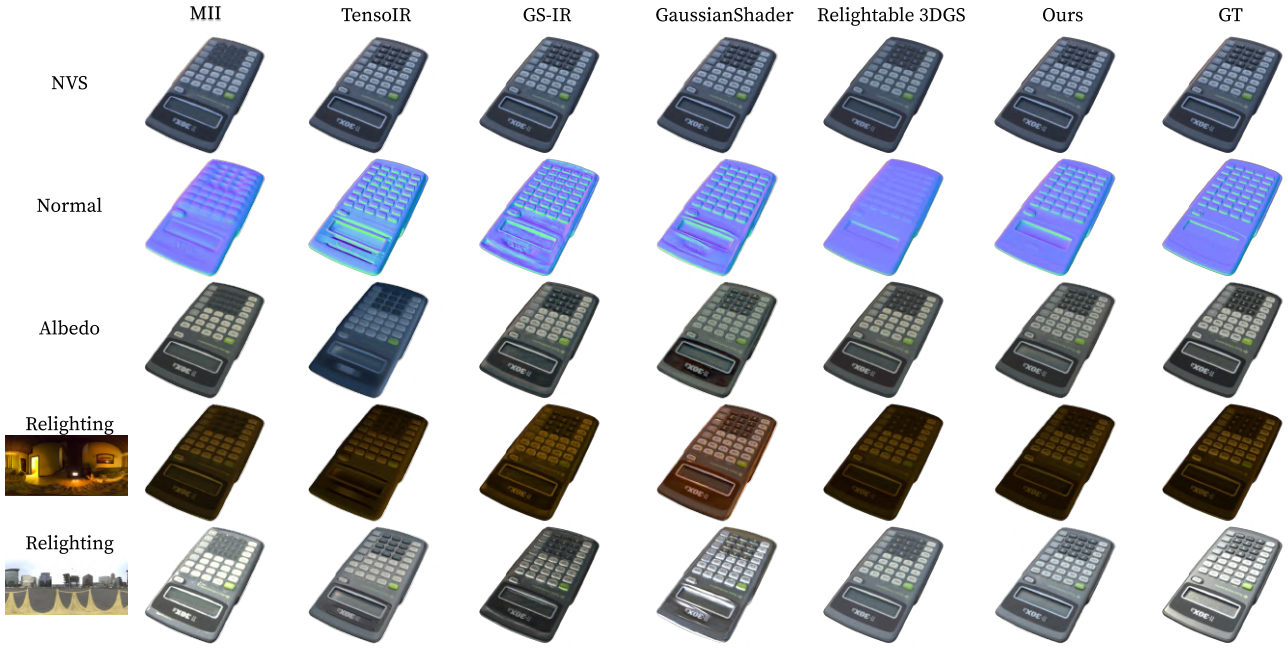


Figure 18. Qualitative comparison of NVS, normal, albedo and relighting on **calculator** of ADT dataset.

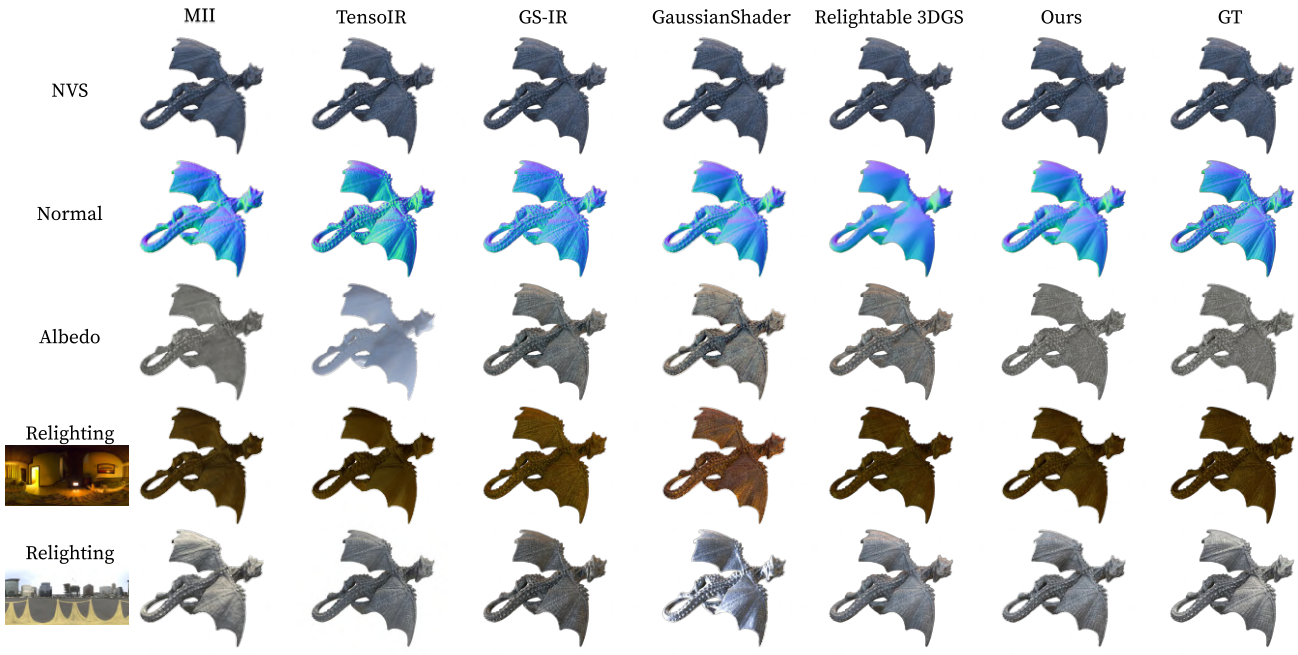


Figure 19. Qualitative comparison of NVS, normal, albedo and relighting on **Gargoyle** of ADT dataset.

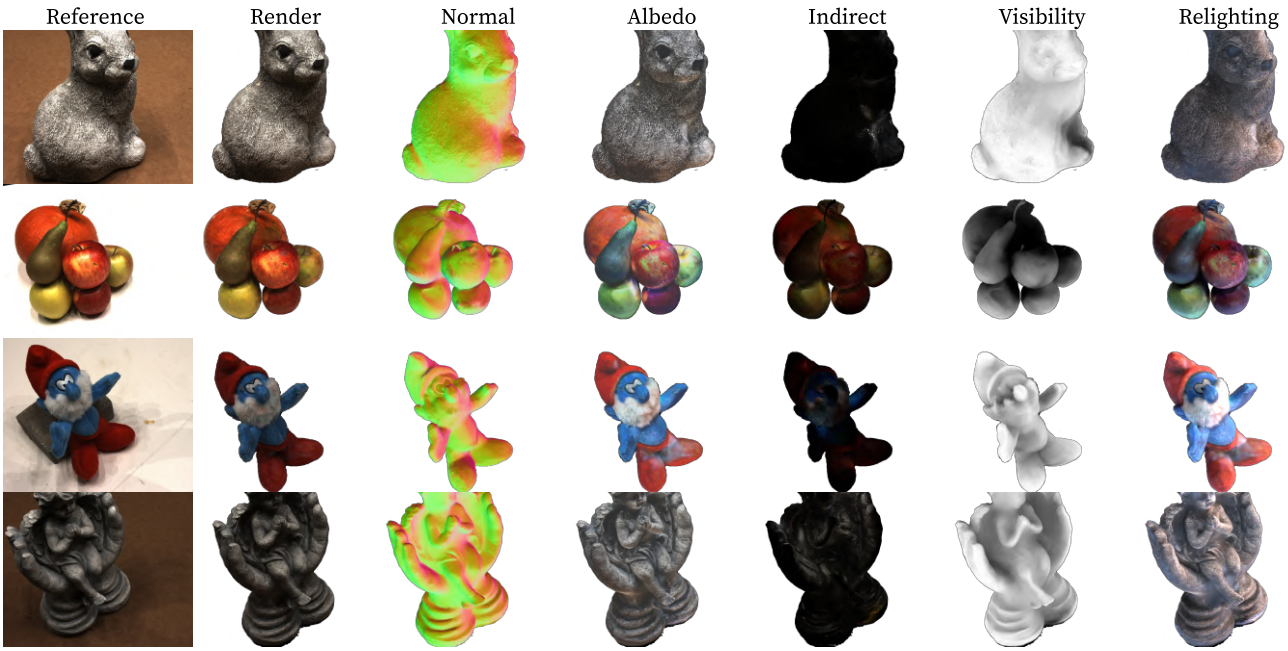


Figure 20. Inverse rendering and relighting results on DTU dataset.

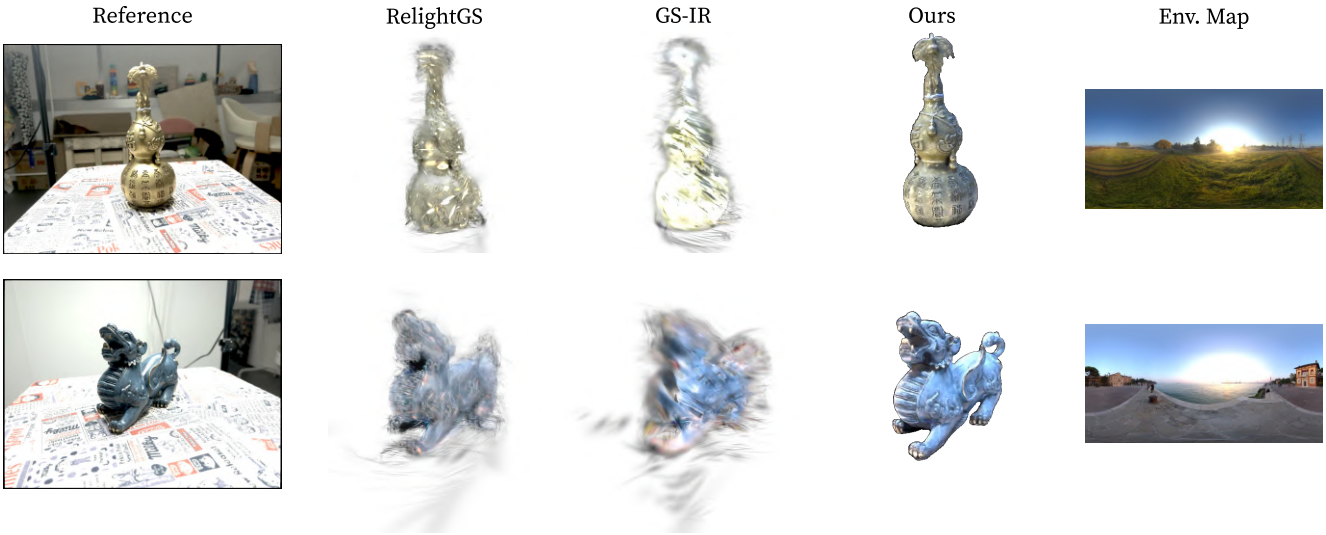


Figure 21. Relighting Qualitative comparison on NeILF++ dataset.

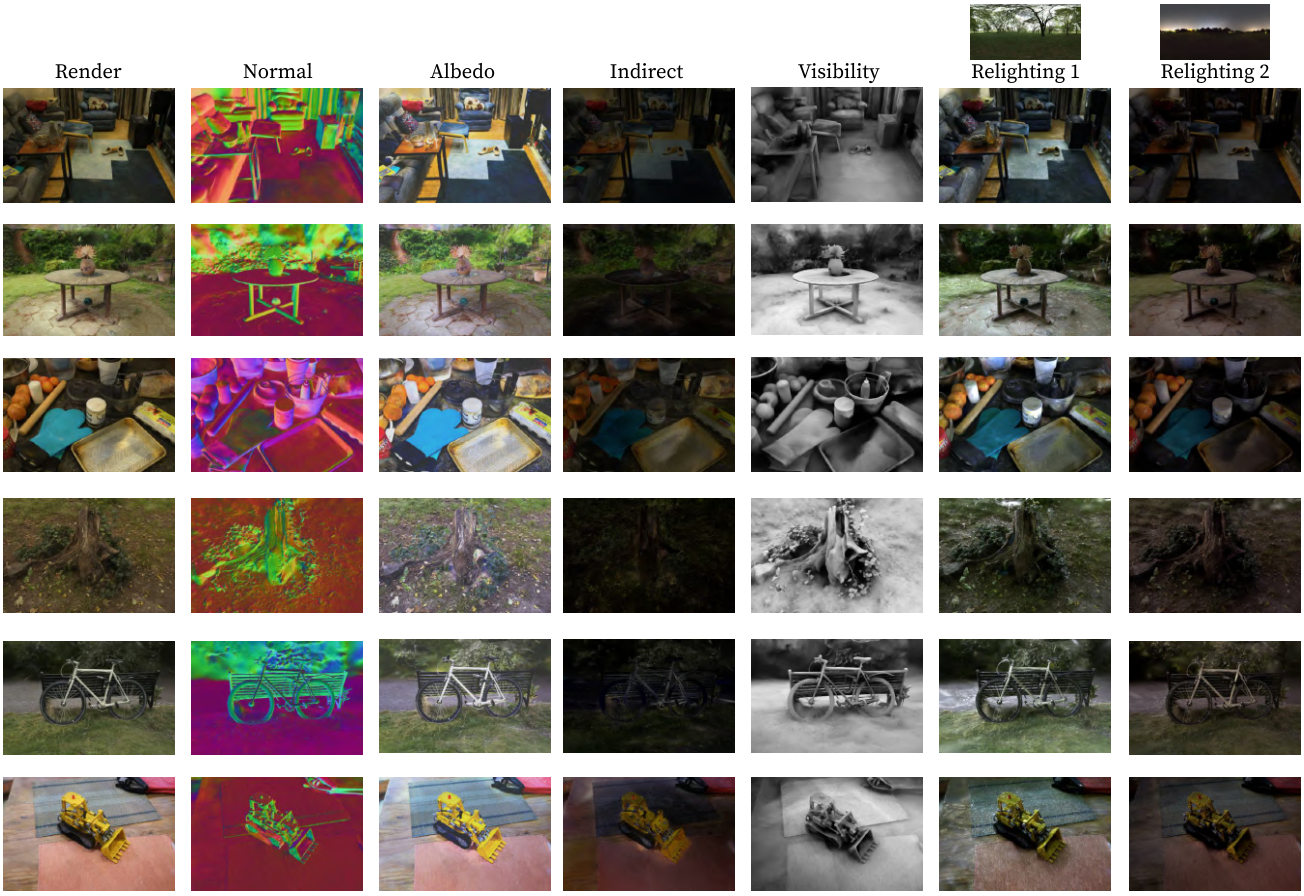


Figure 22. Inverse rendering and relighting results on MipNeRF360 dataset.