GS-IR: 3D Gaussian Splatting for Inverse Rendering

Zhihao Liang[1,*], Qi Zhang[2,*], Ying Feng[2], Ying Shan[2], Kui Jia[3,†]
[1]South China University of Technology, [2] Tencent AI Lab,
[3] School of Data Science, The Chinese University of Hong Kong, Shenzhen
eezhihaoliang@mail.scut.edu.cn, nwpuqzhang@gmail.com,
vonyfeng@gmail.com, yingsshan@tencent.com, kuijia@cuhk.edu.cn

Abstract

*We propose GS-IR, a novel inverse rendering approach based on 3D Gaussian Splatting (3DGS) that leverages forward mapping volume rendering to achieve photorealistic novel view synthesis and relighting results. Unlike previous works that use implicit neural representations and volume rendering (*e.g.* NeRF), which suffer from low expressive power and high computational complexity, we extend 3DGS, a top-performance representation for novel view synthesis, to estimate scene geometry, surface material, and environment illumination from multi-view images captured under unknown lighting conditions. There are two main problems when introducing 3DGS to inverse rendering: 1) 3DGS does not support producing plausible normal natively; 2) forward mapping (*e.g.* rasterization and splatting) cannot trace the occlusion like backward mapping (*e.g.* ray tracing). To address these challenges, our GS-IR proposes an efficient optimization scheme incorporating a depth-derivation-based regularization for normal estimation and a baking-based occlusion to model indirect lighting. The flexible and expressive 3DGS representation allows us to achieve fast and compact geometry reconstruction, photorealistic novel view synthesis, and effective physically-based rendering. We demonstrate the superiority of our method over baseline methods through qualitative and quantitative evaluations of various challenging scenes. The source code is available at https://github.com/lzhnb/GS-IR.*

1. Introduction

Inverse rendering is a long-standing task, seeking to answer the question: "How can we deduce physical attributes (*e.g.* geometry, material, and lighting) of a 3D scene from multi-view images?". This problem is inherently challenging and

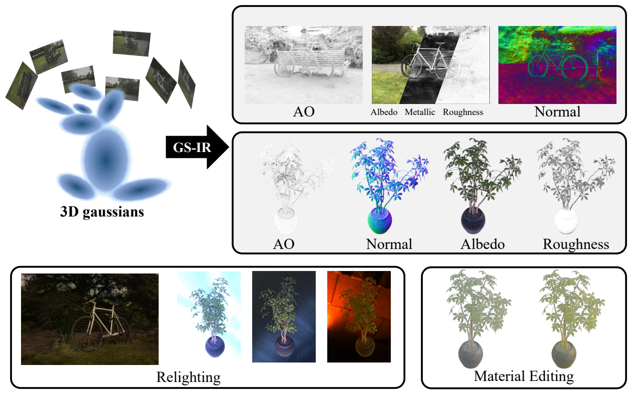


Figure 1. Given multi-view captured images of a complex scene, we propose *GS-IR* (3D **G**aussian **S**platting for **I**nverse **R**endering), which utilizes *3D Gaussian* and forward mapping splatting to recover high-quality physical properties (e.g., normal, material, illumination). This enables us to perform relighting and material editing, resulting in outstanding inverse rendering results. **Better viewed on screen with zoom in**, especially the *remarkable* material decomposition and normal reconstruction of bicycle axle.

ill-posed, particularly when input images are captured in uncontrolled environments with unknown illumination. Recent research [9, 10, 37, 48] has sought to address this issue by employing implicit neural representations akin to NeRF [31] that utilizes multi-layer perceptrons (MLPs). However, current methods incorporating MLP face challenges in terms of their low expressive capacity and high computational demands, which significantly limits the effectiveness and efficiency of inverse rendering, especially when it cannot be rendered at interactive rates.

3D Gaussian Splatting (3DGS) [24] has recently emerged as a promising technique to model 3D static scenes and significantly boost the rendering speed to a real-time level. It makes the scene representation more compact and achieves fast and top performance for novel view synthesis. Introducing it to the inverse rendering pipeline is natural and essential, including geometry reconstruction, materials decomposition, and illumination estimation. Unlike ray trac-

* indicates equal contribution.
†Correspondence to Kui Jia <kuijia@cuhk.edu.cn>.

ing in NeRF, 3DGS produces a set of 3D Gaussians around sparse points. During the 3DGS optimization, the adaptive control of the Gaussian density may lead to loose geometry, making it difficult to estimate accurate scene's normal. Consequently, it is necessary to introduce a well-designed strategy to regularize 3DGS's normal estimation.

Our goal is to use 3D Gaussians as the scene representation for inverse rendering from multi-view images captured under unknown lighting conditions. However, capturing observations under natural illumination often shows complex effects such as soft shadows and interreflections. TensoIR [22] leverages the ray tracing of NeRF to directly model occlusion and indirect illumination. In contrast, 3DGS replaces the ray tracing in the NeRF with differentiable forward mapping volume rendering, which directly projects 3D Gaussians onto the 2D plane. This strategy improves the rendering efficiency but makes it difficult to calculate occlusion. Inspired by the "*Indirect Lighting Cache*" used in real-time rendering [4], we attempt to bake the occlusion into volumes for caching.

In this paper, we present a novel 3D Gaussian-based inverse rendering framework called *GS-IR* (3D Gaussian Splatting for Inverse Rendering) that leverages forward mapping splatting to deduce the physical attributes of a complex scene. To the best of our knowledge, our method is the *first* work to introduce the 3DGS technique for inverse rendering, which can simultaneously estimate scene geometry, materials, and illumination from multi-view images. Our GS-IR addresses two main issues when using 3DGS for inverse rendering. Firstly, we develop an intuitive and well-designed regularization to estimate the scene's normal. Secondly, we use a baking-based method embedded in GS-IR to cache occlusions, obtaining an efficient indirect illumination model. As shown in Fig. 1, our approach can reconstruct high-fidelity geometry and materials of a complex real scene under unknown natural illumination, enabling state-of-the-art rendering of novel view synthesis and additional applications like relighting. Our technical contributions are summarized as follows:

- We present *GS-IR* that models a scene as a set of 3D Gaussians to achieve physically-based rendering and state-of-the-art decomposition results for both objects and scenes;
- We propose an efficient optimization scheme with regularization to concentrate depth gradient around 3DGS and produce reliable normals for GS-IR;
- We develop a baking-based method embedded in GS-IR to handle the occlusion in modeling indirect lighting;

We demonstrate the superiority of our method to baseline methods qualitatively and quantitatively on various challenging scenes, including the TensoIR synthesis dataset [22] and Mip-NeRF 360 real dataset [5].

2. Related Works

Neural Representation Recently, neural rendering techniques, exemplified by Neural Radiance Field (NeRF) [31], have achieved impressive success in addressing visual computing problems, giving rise to numerous neural representations [14, 21, 32, 38, 41, 42] tailored for different tasks [11–13, 18, 28, 29, 35, 45]. The vanilla NeRF models a continuous radiance field implicitly in MLPs, which requires massive repeated queries during training and inference. To address the computational inefficiencies, many neural scene representations are proposed with more discretized geometry proxies such as voxel grids [17, 20, 38], hash grids [32], tri-planes [14] or points [41]. Neural features are stored in a structured manner, allowing for efficient storage and retrieval. The computational cost can thus be significantly reduced by introducing interpolation techniques, however, with an inevitable loss in image quality. 3D Guassians are introduced as an unstructured scene representation to strike a balance between efficiency and quality [24]. With the specially designed tile-based rasterizer for Guassian splats, this method achieves real-time rendering with high quality for novel-view synthesis. In this work, 3D Gaussian representation is combined with the physical-based rendering (PBR) model for inverse rendering.

Inverse Rendering Inverse rendering aims to decompose the image's appearance into the geometry, material, and lighting conditions. Considering the inherent ambiguity between observed images and underlying properties, many methods are proposed with different constrained settings, such as capturing images with fixed lighting and rotating object [16, 40], capturing with moving camera and co-located lighting [7, 8, 30, 34]. Combined with neural representations, inverse rendering models the scene simulating how the light interacts with the neural volume with various material properties, and estimates the lighting and material parameters during optimization [6, 9, 10, 19, 22, 37, 44, 46, 48, 49]. Neural Reflectance Fields [6] assumes a known point light source and represents the scene as a field of volume density, surface normals, and bi-directional reflectance distribution functions(BRDFs) with one bounce direct illumination. NeRV [37] and InvRender [49] extend to arbitrary known lighting conditions and train an additional MLP to model the light visibility. PhySG [44] assumes full light source visibility without shadow simulation, and represents the lighting and scene BRDFs with spherical Guassians for acceleration. TensoIR [22] adopts the efficient TensoRF [14] representation which enables the computation of visibility and indirect lighting by raytracing, while limited to object-level. For modeling surface geometry using point clouds, Fuzzy Metaballs (FMs) [25, 26] offers a great way to render depth from 3D Gaussian using Order Independent Transparency (OIT) and approximate intersection. However, it requires silhouettes as input and struggles to handle intricate geometry (*e.g*. Lego and Ficus) let alone complex scenes. In this work, we propose a 3DGS-based pipeline to recover the geometry, material, and lighting that is available for both objects and unbounded scenes.

3. Preliminary

In this section, we give the technical backgrounds and math symbols that are necessary for the presentation of our proposed method in subsequent sections.

3D Gaussian Splatting (3DGS) [24] is an explicit 3D scene representation in the form of point clouds. Each point is represented as a Gaussian function g that approximates the shape of a bell curve, which is defined as,

$$g(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \tag{1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^3$ is its mean vector, and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ is an anisotropic covariance matrix. The mean vector $\boldsymbol{\mu}$ of a 3D Gaussian is parameterized as $\boldsymbol{\mu} = (\mu_x, \mu_y, \mu_z)$, and the covariance matrix $\boldsymbol{\Sigma}$ is factorized into a scaling matrix $\boldsymbol{S}$ and a rotation matrix $\boldsymbol{R}$ as $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top$. $\boldsymbol{S}$ and $\boldsymbol{R}$ refer to a diagonal matrix $\mathrm{diag}(s_x, s_y, s_z)$ and a rotation matrix constructed from a unit quaternion $\boldsymbol{q}$. Given a viewing transformation with extrinsic matrix $\boldsymbol{T}$ and intrinsic matrix $\boldsymbol{K}$, the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}'$ from the 3D point $\boldsymbol{x}$ to 2D pixel $\boldsymbol{u}$ is defined as,

$$\boldsymbol{\mu}' = \boldsymbol{K}\boldsymbol{T}[\boldsymbol{\mu}, 1]^\top, \quad \boldsymbol{\Sigma}' = \boldsymbol{J}\boldsymbol{T}\boldsymbol{\Sigma}\boldsymbol{T}^\top\boldsymbol{J}^\top, \tag{2}$$

where $\boldsymbol{J}$ is the Jacobian matrix of the affine approximation of the perspective projection. Besides, each Gaussian represents the view-dependent color $\boldsymbol{c}_i$ via a set of coefficients of spherical harmonics (SH), which is then multiplied by opacity α for volume rendering. We finally obtain the color $\hat{\boldsymbol{C}}$ at pixel $\boldsymbol{u}$ based on Eq. (1) and Eq. (2),

$$\hat{\boldsymbol{C}} = \sum_{i\in N} T_i g_i(\boldsymbol{u}|\boldsymbol{\mu}', \boldsymbol{\Sigma}')\alpha_i \boldsymbol{c}_i, \quad T_i = \prod_{j=1}^{i-1}(1 - g_j(\boldsymbol{u}|\boldsymbol{\mu}', \boldsymbol{\Sigma}')\alpha_j), \tag{3}$$

where accumulated transmittance T_i quantifies the probability density of i-th Gaussian at pixel $\boldsymbol{u}$.

The Rendering Equation In GS-IR, we leverage the classic rendering equation to formulate the outgoing radiance of a surface point $\boldsymbol{x}$ with normal $\boldsymbol{n}$:

$$L_o(\boldsymbol{x}, \boldsymbol{v}) = \int_\Omega L_i(\boldsymbol{x}, \boldsymbol{l}) f_r(\boldsymbol{l}, \boldsymbol{v})(\boldsymbol{l}\cdot\boldsymbol{n})d\boldsymbol{l}, \tag{4}$$

Ω denotes the upper hemisphere centered at $\boldsymbol{x}$, $\boldsymbol{l}$ and $\boldsymbol{v}$ denote incident and view directions respectively. $L_i(\boldsymbol{x}, \boldsymbol{l})$ denotes the radiance received at $\boldsymbol{x}$ from $\boldsymbol{l}$. Notably, we follow Cook-Torrance microfacet model [15, 39] and formulate the bidirectional reflectance distribution function (BRDF) f_r as a function of albedo $\boldsymbol{a} \in [0, 1]^3$, metallic $m \in [0, 1]$, and roughness $\rho \in [0, 1]$:

$$f_r(\boldsymbol{l}, \boldsymbol{v}) = \underbrace{(1-m)\frac{\boldsymbol{a}}{\pi}}_{\text{diffuse}} + \underbrace{\frac{DFG}{4(\boldsymbol{n}\cdot\boldsymbol{l})(\boldsymbol{n}\cdot\boldsymbol{v})}}_{\text{specular}}, \tag{5}$$

where microfacet distribution function D, Fresnel reflection F, and geometric shadowing factor G are related to the surface roughness ρ. We use 3D Gaussians to store these material properties in GS-IR.

4. Method

Given a set of calibrated RGB images $\{\boldsymbol{I}_m\}_{m=1}^M$ of a target scene captured from multiple views under static, yet unknown illumination, inverse rendering aims to decompose the scene's intrinsic properties, including normal, materials, and illumination. This decomposition facilitates the recovery and subsequent edition of the target scene. Motivated by the remarkable performance in quality and speed of 3DGS [24], we present a novel framework *GS-IR* consisting of three well-designed stage strategies, as shown in Fig. 2. In the initial stage, we leverage differentiable splatting to optimize 3D Gaussians. Concurrently, we utilize the gradient derived from the rendered depth map to supervise the normal stored in 3D Gaussians (*cf*. Sec. 4.1). In the second stage, we precompute the occlusion based on the learned geometric information (*i.e*. depth and normal) and store it in an efficient spherical harmonics-based architecture to model indirect illumination (*cf*. Sec. 4.2). In the final stage, we combine a differentiable splatting with the physical-based rendering (PBR) pipeline to optimize illumination and material-aware 3D Gaussians (*cf*. Sec. 4.3).

4.1. Normal Reconstruction

During the initial stage, we optimize 3D Gaussians for geometry reconstruction from observed images, denoted as $\mathcal{G}$. The optimized $\mathcal{G}$ functions as a geometric proxy for surface points and their corresponding normals $\boldsymbol{n}$, which are crucial for successful inverse rendering. As highlighted in Sec. 1, generating reasonable normals within the 3DGS-based framework poses a significant challenge. To address this obstacle, we introduce an intuitive strategy that improves depth $\hat{D}$ and leverages the depth gradient to derive pseudo normals $\hat{\boldsymbol{n}}_{\hat{D}} = \nabla_{\text{uv}}\hat{D}$. These pseudo normals then guide the optimization of normals within the 3D Gaussians.

Depth Generation Given a pretrained 3D Gaussians $\mathcal{G}$ and a view designated for rendering, the pixel's shading results in that view can be obtained by Eq. (3). Consequently, it is reasonable to utilize the same volumetric accumulation to compute the depth $\hat{D} = \sum_{i=1}^N T_i\alpha_i d_i$, where d_i denotes the distance from the corresponding 3D Gaussian to the image plane. However, we observed the *floating* problem during volumetric accumulation, unlike the backward mapping volume rendering used in NeRF. During the 3DGS optimization, the adaptive control of the Gaussian density may result in the depth falling in front of the 3D Gaussians, thereby posing challenges in accurately predicting the depth. Specifically, the backward mapping methods can obtain an accurate depth by considering only peak samples,

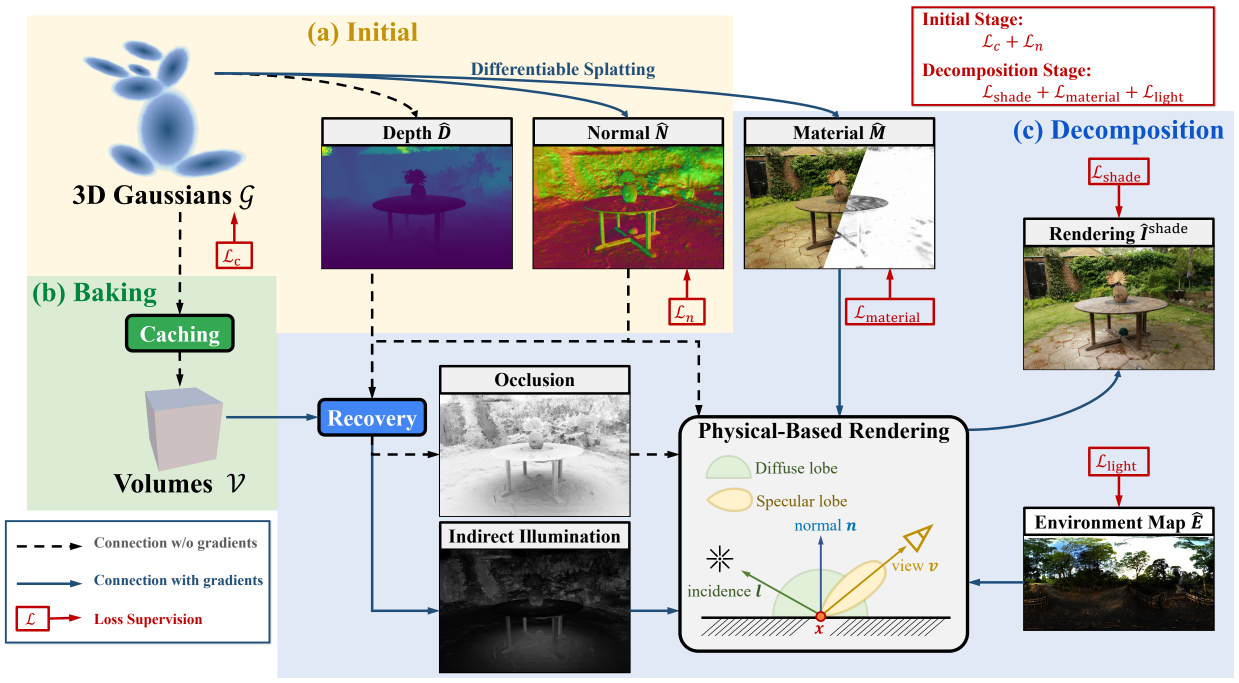

(a) Initial
Differentiable Splatting
3D Gaussians $\mathcal{G}$
$\mathcal{L}_c$
Depth $\hat{D}$
Normal $\hat{N}$
$\mathcal{L}_n$
Material $\hat{M}$
$\mathcal{L}_{\text{material}}$
Initial Stage: $\mathcal{L}_c + \mathcal{L}_n$
Decomposition Stage: $\mathcal{L}_{\text{shade}} + \mathcal{L}_{\text{material}} + \mathcal{L}_{\text{light}}$
(c) Decomposition
$\mathcal{L}_{\text{shade}}$
Rendering $\hat{I}^{\text{shade}}$
(b) Baking
Caching
Volumes $\mathcal{V}$
Recovery
Occlusion
Indirect Illumination
Physical-Based Rendering
Diffuse lobe
Specular lobe
normal $\boldsymbol{n}$
view $\boldsymbol{v}$
incidence $\boldsymbol{l}$
$\boldsymbol{x}$
$\mathcal{L}_{\text{light}}$
Environment Map $\hat{E}$
Connection w/o gradients
Connection with gradients
Loss Supervision

Figure 2. **GS-IR Pipeline**. We propose a novel GS-based inverse rendering framework, called GS-IR, to reconstruct scene geometry, materials, and unknown natural illumination from multi-view captured images. Our GS-IR consists of three well-designed stage strategies using 3D Gaussian and differentiable forward mapping splatting to achieve physical-based rendering. In our approach, the Gaussian stores not only the basic 3DGS information but also the normal and material properties, enhancing its capabilities for inverse rendering tasks.

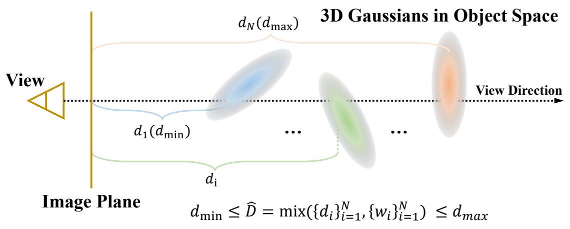

3D Gaussians in Object Space
$d_N(d_{\max})$
View
View Direction
$d_1(d_{\min})$
...
d_i
Image Plane
$d_{\min} \leq \hat{D} = \text{mix}(\{d_i\}_{i=1}^N, \{w_i\}_{i=1}^N) \leq d_{max}$

Figure 3. **Depth Illustration**. By considering the depth as a linear interpolation of the distances from 3D Gaussians to the image plane, and ensuring it lies between the minimum and maximum distance, our method could produce accurate depth.

that is $\hat{D} = d_{i^*}$, where $i^* = \arg\max_i T_i\alpha_i$. However, for 3DGS, a typical forward mapping method, the peak selection results in *disc aliasing* within 3DGS. To overcome this limitation, we consider that depth $\hat{D}$ must be between the minimum and maximum distance of 3D Gaussians to the image plane, as illustrated in Fig. 3. We then treat the depth as a linear interpolation of the distances from 3D Gaussians to the image plane:

$$\hat{D} = \sum_{i=1}^{N} \hat{w}_i d_i, \quad \hat{w}_i = \frac{T_i\alpha_i}{\sum_{i=1}^N T_i\alpha_i}. \tag{6}$$

Normal Derivation While the accurate prediction of depth within Gaussians provides better guidance for the normal reconstruction, directly using depth gradient to produce normals has two limitations that still cannot meet the requirement for effective inverse rendering. First, the depth gradient estimation is highly sensitive to noise, making the predicted normal often extremely noisy; Second, the normals derived individually from each view's depth map do not satisfy multi-view consistency. To address these issues, we use Gaussian $\mathcal{G}$ as a proxy for normal estimation instead of directly from the depth gradient. Benefiting from the efficiency of 3DGS, we obtain the depth $\hat{D}$ and normals $\hat{\boldsymbol{n}}$ of the observed view after performing a single rendering pass. We then tie these predicted pseudo normal to the underlying depth gradient normal $\hat{\boldsymbol{n}}_{\hat{D}}$ using a simple penalty:

$$\mathcal{L}_{n\text{-}p} = \|\hat{\boldsymbol{n}} - \hat{\boldsymbol{n}}_{\hat{D}}\|, \tag{7}$$

where $\hat{\boldsymbol{n}} = \sum_{i=1}^N T_i\alpha_i\boldsymbol{n}_i$, and $\boldsymbol{n}_i$ is the normal stored in the 3D Gaussian. Secondly, unlike the MLP-based normal estimation [22] acting MLP as a low-pass filter, the predicted normal of Gaussian $\mathcal{G}$ is rough, so smoothness regularization should be included. We introduce the TV term TV_{normal} to smooth the predicted normal $\hat{\boldsymbol{n}}$. For more details, please refer to the *supplement*.

In optimization of the first stage, we optimize 3D Gaussians $\mathcal{G}$ (storing SH coefficients for view-dependent color $\boldsymbol{c}$, opacity α, and normal $\hat{\boldsymbol{n}}$) by using the color reconstruction loss $\mathcal{L}_c$, which is the same as 3DGS [24], and the proposed normal loss $\mathcal{L}_n$,

$$\mathcal{L}_n = \mathcal{L}_{n\text{-}p} + \lambda_{n\text{-}TV}\, TV_{\text{normal}}. \tag{8}$$

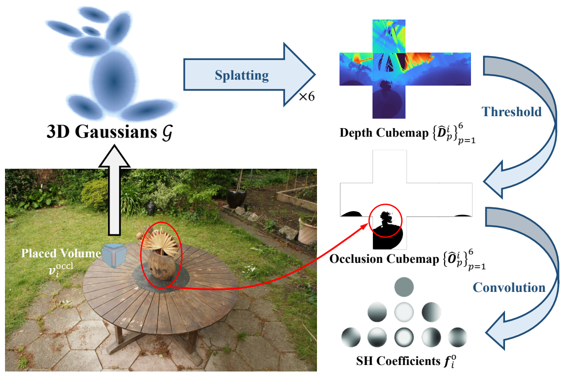

Splatting
×6
3D Gaussians $\mathcal{G}$
Depth Cubemap $\{\hat{D}_p^i\}_{p=1}^6$
Threshold
Placed Volume v_i^{occl}
Occlusion Cubemap $\{\hat{O}_p^i\}_{p=1}^6$
Convolution
SH Coefficients f_i^{o}

Figure 4. **Baking**. We employ the spherical harmonics (SH) architecture to bake occlusion volumes for modeling indirect illumination. For each grid of occlusion volumes, we initially use 3D Gaussians to compute the depth cubemap by performing six forward mapping splatting passes. Next, we convert the depth cubemap into a binary occlusion cubemap based on a distance threshold. Finally, the occlusion cubemap is baked as SH coefficients, enabling efficient interpolation of the occlusion cubemap at any point within the scene.

4.2. Indirect Illumination Modeling

Drawing inspiration from the successful implementation of precomputation techniques in the video game industry (*e.g.* Irradiance Volume in Blender [3], Lightmass Volume in Unreal [1], and Light Probes in Unity [2]), we introduce spherical harmonics (SH) architectures to store occlusion information and model indirect illumination.

Given the optimized 3D Gaussians $\mathcal{G}$ from one stage (*cf*. Sec. 4.1), we freeze $\mathcal{G}$ and regularly place occlusion volumes $\mathcal{V}^{\mathrm{occl}}$ in the 3D space. For each volume $\boldsymbol{v}_i^{\mathrm{occl}} \subset \mathcal{V}^{\mathrm{occl}}$, we then cache the occlusion in the form of SH coefficients $\boldsymbol{f}_i^{\mathrm{o}}$. Consequently, the formula of the occlusion $O(\cdot)$ of $\boldsymbol{v}_i^{\mathrm{occl}}$ with respect to the direction (θ, ϕ) is expressed as:

$$O(\theta,\phi) = \sum_{l=0}^{deg} \sum_{m=-l}^{l} \boldsymbol{f}_{i(lm)}^{\mathrm{o}} Y_{lm}(\theta,\phi), \tag{9}$$

where deg denotes the degree of SH, and $\{Y_{lm}(\cdot)\}$ is a set of real basis of SH.

As discussed in Sec. 4.1, the 3DGS technique employs a forward mapping approach that projects 3D points to the 2D plane, in contrast to the backward mapping volume rendering utilized in NeRF, which means it cannot use ray marching to calculate occlusions. To precompute the SH coefficients $\boldsymbol{f}_i^{\mathrm{o}}$ of occlusion volume $\boldsymbol{v}_i^{\mathrm{occl}}$, we obtain the depth cubemap $\{\hat{\boldsymbol{D}}_p^i\}_{p=1}^6$ by performing six times rendering passes, once for each face of the cubemap. We then convert it into a binary occlusion cubemap $\{\hat{\boldsymbol{O}}_p^i\}_{p=1}^6$ based on a manually set distance threshold.

Finally, we convolve the the occlusion cubemap $\{\hat{\boldsymbol{O}}_p^i\}_{p=1}^6$ using SH bases and get the SH coefficients $\boldsymbol{f}_i^{\mathrm{o}}$:

$$\begin{aligned} \boldsymbol{f}_{i(lm)}^{\mathrm{o}} &= \int_{S^2} \hat{\boldsymbol{O}}(\boldsymbol{\omega}) Y_{lm}(\boldsymbol{\omega}) d\boldsymbol{\omega} \\ &= \int_0^{2\pi}\int_0^{\pi} \sin\theta\, \hat{\boldsymbol{O}}(\theta,\phi) Y_{lm}(\theta,\phi) d\theta d\phi, \end{aligned} \tag{10}$$

where S^2 denotes the unit sphere, and $\hat{\boldsymbol{O}}(\theta,\phi)$ denotes the occlusion query from the occlusion cubemap $\{\hat{\boldsymbol{O}}_p^i\}_{p=1}^6$. Note that we numerically calculate the convolution in Eq. (10) in parallel. The caching process is shown in Fig. 4.

To handle the indirect illumination in occlusion regions, we also maintain illumination volumes $\mathcal{V}^{\mathrm{illu}}$ to cache the indirect illumination. Similar to the caching process of occlusion volumes, the caching target of illumination volumes changes from the occlusion cubemap $\{\hat{\boldsymbol{O}}_p^i\}_{p=1}^6$ to the captured environment cubemap $\{\hat{\boldsymbol{I}}_p^i\}_{p=1}^6$. These cubemaps can be obtained simultaneously by conducting six rendering passes. For more details, please refer to the *supplement*.

4.3. Intrinsic Decomposition

In the final stage, we employ differentiable splatting in conjunction with a PBR pipeline to accomplish the intrinsic decomposition. According to Eq. (5), the rendering equation Eq. (4) is rewritten as diffuse L_{d} and specular L_{s} components:

$$\begin{aligned} L_o(\boldsymbol{x},\boldsymbol{v}) &= \int_\Omega \left[(1-m)\frac{\boldsymbol{a}}{\pi} + \frac{DFG}{4(\boldsymbol{n}\cdot\boldsymbol{l})(\boldsymbol{n}\cdot\boldsymbol{v})}\right] L_i(\boldsymbol{x},\boldsymbol{l})(\boldsymbol{l}\cdot\boldsymbol{n})d\boldsymbol{l} \\ L_{\mathrm{d}} &= (1-m)\frac{\boldsymbol{a}}{\pi}\int_\Omega L_i(\boldsymbol{x},\boldsymbol{l})(\boldsymbol{l}\cdot\boldsymbol{n})d\boldsymbol{l} \\ L_{\mathrm{s}} &= \int_\Omega \frac{DFG}{4(\boldsymbol{n}\cdot\boldsymbol{l})(\boldsymbol{n}\cdot\boldsymbol{v})} L_i(\boldsymbol{x},\boldsymbol{l})(\boldsymbol{l}\cdot\boldsymbol{n})d\boldsymbol{l}. \end{aligned} \tag{11}$$

In GS-IR, we adopt an image-based lighting (IBL) model and split-sum approximation [23] to tackle the intractable integral. To calculate the diffuse component L_{d}, the illumination I_{d} is defined as:

$$\begin{aligned} I_{\mathrm{d}}(\boldsymbol{x}) &= \int_\Omega L_i(\boldsymbol{x},\boldsymbol{l})(\boldsymbol{l}\cdot\boldsymbol{n})d\boldsymbol{l} \\ &= \int_{\Omega_{\mathrm{Vis}}} L_i^{\mathrm{dir}}(\boldsymbol{x},\boldsymbol{l})(\boldsymbol{l}\cdot\boldsymbol{n})d\boldsymbol{l} + \int_{\Omega_{\mathrm{Occl}}} L_i^{\mathrm{indir}}(\boldsymbol{x},\boldsymbol{l})(\boldsymbol{l}\cdot\boldsymbol{n})d\boldsymbol{l} \\ &\approx (1-\mathrm{O}(\boldsymbol{x}))\, I_{\mathrm{d}}^{\mathrm{dir}}(\boldsymbol{x}) + \mathrm{O}(\boldsymbol{x}) I_{\mathrm{d}}^{\mathrm{indir}}(\boldsymbol{x}), \end{aligned} \tag{12}$$

where the first component indicates direct illumination and the second is indirect illumination. Notably, our baking-based indirect illumination model enables us to calculate the occlusion and illumination online. This means that our GS-IR achieves intrinsic decomposition while maintaining real-time rendering performance. For the specular component L_{s}, we follow split-sum approximation and treat the integral as two separate integrals:

$$\begin{aligned} L_{\mathrm{s}} &= \int_\Omega \frac{DFG}{4\,\langle\boldsymbol{n}\cdot\boldsymbol{l}\rangle\,\langle\boldsymbol{n}\cdot\boldsymbol{v}\rangle} L_i(\boldsymbol{l})\,\langle\boldsymbol{l}\cdot\boldsymbol{n}\rangle\, d\boldsymbol{l} \\ &\approx \underbrace{\int_\Omega \frac{DFG}{4\,\langle\boldsymbol{n}\cdot\boldsymbol{l}\rangle\,\langle\boldsymbol{n}\cdot\boldsymbol{v}\rangle}\,\langle\boldsymbol{l}\cdot\boldsymbol{n}\rangle\, d\boldsymbol{l}}_{\text{Environment BRDF - } R} \underbrace{\int_\Omega D\, L_i(\boldsymbol{l})\,\langle\boldsymbol{l}\cdot\boldsymbol{n}\rangle\, d\boldsymbol{l}}_{\text{Pre-Filtered Environment Map - } I_s}, \end{aligned} \tag{13}$$

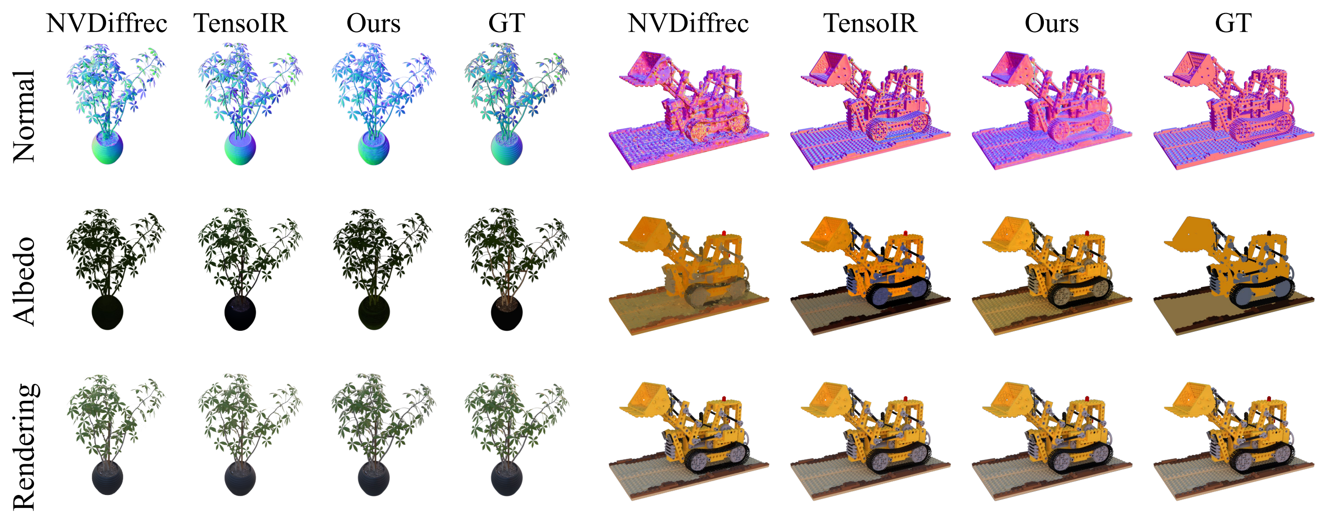


Figure 5. **Qualitative comparison on TensoIR Synthetic**. We visualize the estimated normal, albedo, and rendering results of our GS-IR and baseline methods on two scenes. By utilizing the efficient 3D Gaussian representation and a robust tile-based rasterizer, GS-IR achieves rapid convergence and supports real-time rendering. This performance advantage underscores the effectiveness of our method in addressing complex inverse rendering tasks, thereby surpassing existing state-of-the-art approaches. (For albedo reconstruction results, we follow NeRFactor [48] and scale each RGB channel by a global scalar.)

where both R and I_s can be precomputed in advance and stored in look-up tables. With Eq. (12) and Eq. (13), the rendering results of Eq. (11) can be represented as:

$$\begin{aligned} L_o(\boldsymbol{x},\boldsymbol{v}) &= L_d + L_s \\ &\approx (1-m)\frac{\boldsymbol{a}}{\pi}\left[(1-\mathrm{O}(\boldsymbol{x}))\, I_d^{\text{dir}}(\boldsymbol{x}) + \mathrm{O}(\boldsymbol{x}) I_d^{\text{indir}}(\boldsymbol{x})\right] + R I_s. \end{aligned} \tag{14}$$

For intrinsic decomposition, we optimize the material $\hat{\boldsymbol{M}}$ (*i.e.* albedo $\boldsymbol{a}$, metallic value m, and roughness ρ) stored in 3D Gaussians $\mathcal{G}$, environment map $\hat{\boldsymbol{E}}$, and illumination volumes $\mathcal{V}^{\text{illu}}$ by minimizing the decomposition loss $\mathcal{L}_d$:

$$\mathcal{L}_d = \underbrace{\left\| \boldsymbol{I} - \hat{\boldsymbol{I}}^{\text{shade}}(\hat{\boldsymbol{M}}, \hat{\boldsymbol{E}}, \mathcal{V}^{\text{illu}}) \right\|}_{\mathcal{L}_{\text{shade}}} + \underbrace{\lambda_{\boldsymbol{M}}\ TV_{\text{mat}}}_{\mathcal{L}_{\text{material}}} + \underbrace{\lambda_{\boldsymbol{E}}\ TV_{\text{light}}}_{\mathcal{L}_{\text{light}}}, \tag{15}$$

where $\mathcal{L}_{\text{shade}}$ indicate the shade loss. $\hat{\boldsymbol{I}}^{\text{shade}}(\hat{\boldsymbol{M}}, \hat{\boldsymbol{E}}, \mathcal{V}^{\text{illu}})$ is the recovered image that uses the PBR pipeline defined by Eq. (14).Please refer to the *supplment* for more details about the material TV loss $\mathcal{L}_{\text{material}}$ and lighting TV loss $\mathcal{L}_{\text{light}}$.

5. Experiments

Dataset & Metrics We conduct experiments using benchmark datasets of TensoIR Synthetic [22] and Mip-NeRF 360 [5] for decompositing both objects and scenes. They contain 4 objects with reference materials and 7 publicly available scenes, respectively. To verify the efficacy of our normal reconstruction, we evaluate the normal quality on the TensoIR Synthetic [22] dataset using mean angular error (MAE). We further assess our reconstructed albedo quality on this synthetic dataset. More generally, we evaluate the synthesized novel view on both datasets in terms of Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [47]. Note that albedo quality assessment uses the same metrics as novel view synthesis.

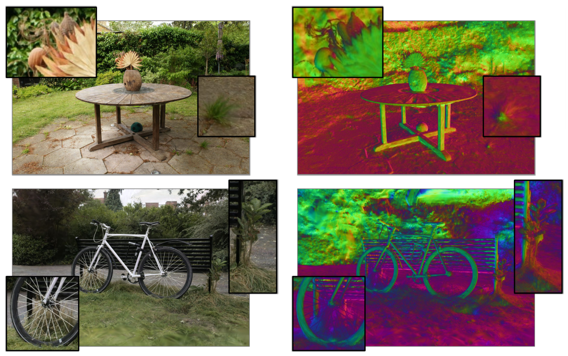


Figure 6. **Novel view synthesis results on Mip-NeRF 360.** GS-IR can reconstruct scene details including geometric normals and high-frequency appearance, rendering high-fidelity appearance and recovering fine geometric details such as those on leaves and bicycle axles. **Better viewed on screen with zoom in**.

5.1. Comparisons

We conduct a comprehensive comparison against state-of-the-art neural field-based inverse rendering methods on the public TensoIR Synthetic dataset [22]. All the methods utilize multi-view images captured under unknown lighting conditions. Our evaluation encompasses normal quality

Method	Normal MAE ↓	Novel View Synthesis PSNR ↑	Novel View Synthesis SSIM ↑	Novel View Synthesis LPIPS ↓	Albedo PSNR ↑	Albedo SSIM ↑	Albedo LPIPS ↓	Relight PSNR ↑	Relight SSIM ↑	Relight LPIPS ↓	Runtime
NeRFactor [48]	6.314	24.679	0.922	0.120	25.125	0.940	0.109	23.383	0.908	0.131	> 100 hrs
InvRender [49]	5.074	27.367	0.934	0.089	27.341	0.933	0.100	23.973	0.901	0.101	15 hrs
NVDiffrec [33]	6.078	30.696	0.962	0.052	29.174	0.908	0.115	19.880	0.879	0.104	< 1 hr
TensoIR [22]	4.100	35.088	0.976	0.040	29.275	0.950	0.085	28.580	0.944	0.081	5 hrs
Ours	4.948	35.333	0.974	0.039	30.286	0.941	0.084	24.374	0.885	0.096	< 1 hr

Table 1. **Quantitative Comparison on TensoIR Synthetic dataset.** Our method outperforms baseline methods in terms of novel view synthesis and albedo quality, showcasing the effectiveness of material decomposition and PBR rendering. This is particularly noteworthy considering that our normal reconstruction is slightly inferior to TensoIR. In terms of relighting performance, we rank second, trailing only behind TensoIR. Importantly, the average training time of our GS-IR is accelerated by a factor of 5x, making its performance acceptable and further demonstrating the effectiveness of our approach in handling complex inverse rendering tasks.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Runtime ↓
NeRF++ [43]	25.112	0.696	0.375	≈ 20h
Plenoxels [17]	23.079	0.625	0.462	≈ 30m
INGP-Base [32]	25.303	0.671	0.371	≈ **5m**
INGP-Big [32]	25.587	0.699	0.331	≈ 8m
Mip-NeRF 360 [32]	27.569	0.793	0.234	≈ 48h
3DGS [24]	27.21	0.815	0.214	≈ 35m
Ours	25.381	0.757	0.267	≈ **45m**

Table 2. **Quantitative Comparison on Mip-NeRF 360.** The results show that our inverse rendering approach even surpasses some NeRF variants dedicated to novel view synthesis.

(measured by MAE), novel view synthesis, albedo fidelity, relighting effects (measured by PSNR, SSIM, and LPIPS), and efficiency. Tab. 1 summarizes the quantitative comparisons on the synthesis dataset. Our method achieves superior performance in novel view synthesis and albedo quality compared to the baseline methods, demonstrating the effectiveness of material decomposition and PBR rendering, particularly given that our normal reconstruction is slightly inferior to TensoIR. Our relighting performance ranks second, only behind TensoIR. Notably, the average training time of our GS-IR is accelerated by a factor of 5x, making its performance acceptable and demonstrating the effectiveness of our approach in handling complex inverse rendering tasks. We also include qualitative comparisons in Fig. 5, which show that our GS-IR produces reasonable albedo and photorealistic renderings that are closer to the ground truth than most methods.

Meanwhile, owing to our more efficient and compact representation with powerful expressiveness, our method showcases remarkable performance on complex real unbounded scenes [5]. Tab. 2 presents the quantitative comparisons on the real dataset. Fig. 6 demonstrates the normal reconstruction and novel view synthesis on the real dataset. Our method renders a high-fidelity appearance and recovers fine geometric details, such as those on the leaves and bicycle axil. In summary, by leveraging the efficient 3D Gaussian representation and a powerful tile-based rasterizer, GS-IR achieves fast convergence and supports real-time rendering. This performance advantage highlights the effectiveness of our method in handling complex inverse rendering tasks, outperforming existing state-of-the-art approaches.

Method	TensoIR Synthetic [22] Normal MAE ↓	TensoIR Synthetic [22] PSNR ↑	TensoIR Synthetic [22] SSIM ↑	TensoIR Synthetic [22] LPIPS ↓	Mip-NeRF 360 [5] PSNR ↑	Mip-NeRF 360 [5] SSIM ↑	Mip-NeRF 360 [5] LPIPS ↓
Vol. Accum.	16.347	25.756	0.855	0.131	22.052	0.610	0.394
Peak Selec.	9.466	28.750	0.927	0.084	23.093	0.719	0.317
Linear Interp.	6.218	28.983	0.939	0.066	23.119	0.707	0.319
Vol. Accum.†	9.315	30.091	0.940	0.071	24.664	0.747	0.277
Peak Selec.†	7.986	31.064	0.950	0.060	25.143	0.753	0.281
Linear Interp.†	**4.948**	**35.333**	**0.974**	**0.039**	**25.381**	**0.757**	**0.267**

Table 3. **Analyses on the impact of different depth generation strategies on normals.** Methods without † marks directly use the normals derived from the depth map; Methods marked with † use depth derivation to optimize the normals stored in 3D Gaussians.

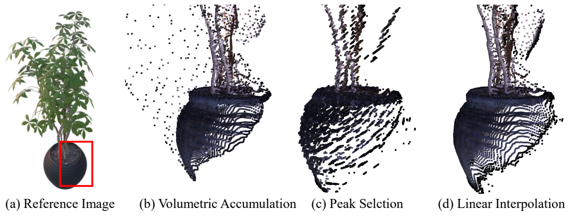


Figure 7. **Visual comparison of depth produced by different strategies.** The linear interpolation adopted in GS-IR overcomes the *floating* problem and *disc aliasing*.

Method	TensoIR Synthetic [22] PSNR ↑	TensoIR Synthetic [22] SSIM ↑	TensoIR Synthetic [22] LPIPS ↓	Mip-NeRF 360 [5] PSNR ↑	Mip-NeRF 360 [5] SSIM ↑	Mip-NeRF 360 [5] LPIPS ↓
w/o occlusion	34.997	0.962	0.041	25.060	0.753	0.270
w/o indirect illum.	35.186	0.965	0.044	24.898	0.749	0.272
Ours	35.333	0.974	0.039	25.381	0.757	0.267

Table 4. **Analyses on the occlusion and indirect illumination.** Physically modeling indirect illumination improves the inverse rendering of objects and scenes.

5.2. Ablation Studies

We initially introduce the 3DGS technique for inverse rendering in GS-IR and propose depth-derivation-based normal regularization and a baking-based method to address the challenges encountered during the process. To evaluate the efficacy of our proposed schemes, we design elaborate experiments on both TensoIR Synthetic [22] and Mip-NeRF 360 [5] datasets, providing comprehensive insights into the effectiveness of our approach in handling complex inverse rendering tasks. Below is the detailed ablation study

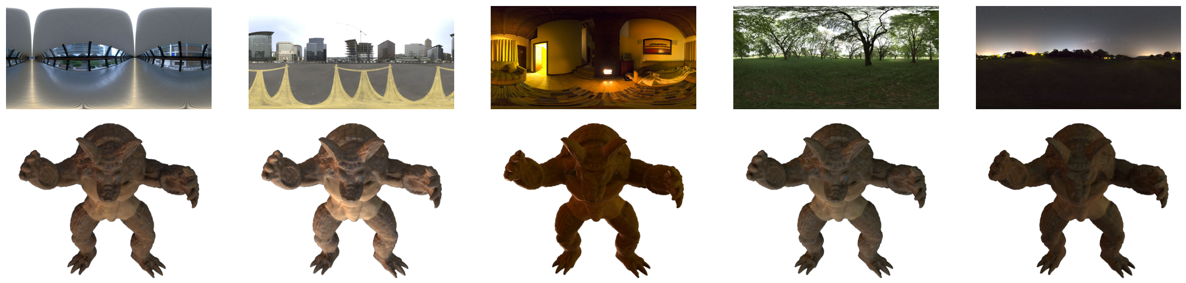


Figure 8. **Relighting Visualization.** We perform relighting experiments on both synthetic and real scenes using the recovered geometry, material, and illumination properties from our GS-IR method. We test our method under different lighting conditions and directions.

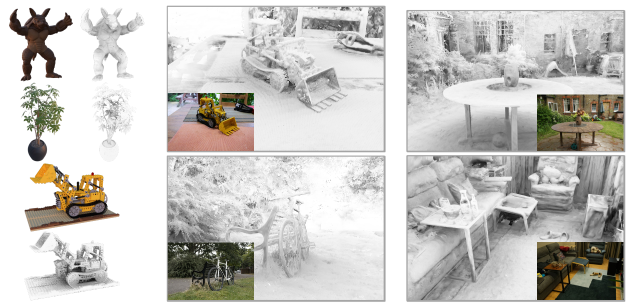


Figure 9. **Ambient Occlusion Visualization.** The visualization highlights the intricate shadowing and occlusion details captured by our GS-IR method, emphasizing the performance of our approach in modeling indirect illumination.

on Normal Regularization and Indirect Illumination.

Analysis on the Normal Regularization Reliable normal estimation is critical for conducting inverse rendering. To this end, we present depth-derivation-based regularization to facilitate 3D Gaussian-based normal estimation as stated in Sec. 4.1. In this section, we explore the impact of different acquisition schemes on the final normal quality and inverse rendering results. The quantitative results shown in Tab. 3 demonstrate that using 3D Gaussians as a normal proxy and adopting the linear interpolation strategy significantly improves the normal estimation and inverse rendering results. In addition, Fig. 7 qualitatively shows that conducting volumetric accumulation results in the *floating* problem (*cf*. Fig. 7 (b)). Despite peak selection overcomes this problem, it introduces *disc aliasing* (*cf*. Fig. 7 (c)). Compared with them, the linear interpolation adopted in GS-IR robustly produces accurate depth (*cf*. Fig. 7 (d)).

Analysis on the Indirect Illumination To demonstrate the effectiveness of our indirect illumination model, we compare our method with two variants: a model without occlusion volume (w/o occlusion) and a model without indirect illumination (w/o indirect illum.). The quantitative comparisons in Tab. 4 indicate that each component is crucial for estimating accurate material decomposition and generating photorealistic rendering results. Additionally, Fig. 9 showcases the ambient occlusion visualization in both synthetic and real scenes. The visualization highlights the intricate shadowing and occlusion details captured by our GS-IR method, emphasizing the performance of our approach in modeling indirect illumination. This analysis further supports the effectiveness of using occlusion volume and introducing indirect illumination in enhancing the decomposition capabilities of our GS-IR.

5.3. Application

We perform relighting experiments using the recovered geometry, material, and illumination from our GS-IR method. We test GS-IR under different lighting conditions and directions, observing how the reconstructed scene responds to the changes in lighting. The results of these experiments demonstrate that our GS-IR method can effectively handle relighting applications, producing photorealistic renderings under various lighting conditions. More results can be found in the *supplement*.

6. Conclusion

We present GS-IR, a novel inverse rendering approach based on 3D Gaussian Splatting (3DGS), which employs forward mapping volume rendering to achieve photorealistic novel view synthesis and relighting results. Our GS-IR proposes an optimization scheme with depth-derivation-based regularization for normal estimation and a baking-based occlusion to model indirect lighting. These components are eventually employed to decompose material and illumination. Our extensive experiments demonstrate the effectiveness of GS-IR in achieving state-of-the-art inverse rendering results, surpassing previous neural methods in terms of both reconstruction quality and efficiency.

Limitation Spherical Harmonics (SH) is only suitable for representing low-frequency, and we only use the occlusion represented by SH to model the diffuse term of indirect illumination. Modeling the specular term of indirect illumination remains a limitation of GS-IR, and has been a challenging problem in computer graphics. We believe it would be valuable to address this limitation in future work and suggest screen space global illumination (SSGI) techniques.

References

[1] Unreal engine. https://www.unrealengine.com/. 5

[2] Unity. https://unity.com/. 5

[3] Blender. https://www.blender.org/. 5

[4] Tomas Akenine-Mo, Eric Haines, Naty Hoffman, et al. Real-time rendering. 2018. 2

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 6, 7, 3

[6] 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. *arXiv preprint arXiv:2008.03824*, 2020. 2

[7] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 294–311. Springer, 2020. 2

[8] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David Kriegman, and Ravi Ramamoorthi. Deep 3d capture: Geometry and reflectance from sparse multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5960–5969, 2020. 2

[9] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021. 1, 2

[10] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan Barron, and Hendrik Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. *Advances in Neural Information Processing Systems*, 34: 10691–10704, 2021. 1, 2

[11] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2

[12] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021.

[13] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 2

[14] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2

[15] Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (ToG)*, 1(1):7–24, 1982. 3

[16] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Transactions on Graphics (TOG)*, 33(6):1–12, 2014. 2

[17] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2, 7, 3

[18] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2

[19] 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. 2

[20] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 2

[21] Zhangjin Huang, Zhihao Liang, Haojie Zhang, Yangkai Lin, and Kui Jia. Sur2f: A hybrid representation for high-quality and efficient surface reconstruction from multi-view images. *arXiv preprint arXiv:2401.03704*, 2024. 2

[22] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensoir: Tensorial inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2023. 2, 4, 6, 7

[23] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 4(3):1, 2013. 5

[24] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 1, 2, 3, 4, 7

[25] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *European Conference on Computer Vision*, pages 596–614. Springer, 2022. 2

[26] Leonid Keselman and Martial Hebert. Flexible techniques for differentiable rendering with 3d gaussians. *arXiv preprint arXiv:2308.14737*, 2023. 2

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[28] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition, pages 6498–6508, 2021. 2

[29] Zhihao Liang, Zhangjin Huang, Changxing Ding, and Kui Jia. Helixsurf: A robust and efficient neural implicit surface learning of indoor scenes with iterative intertwined regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13165–13174, 2023. 2

[30] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable monte carlo rendering. In *Computer Graphics Forum*, pages 101–113. Wiley Online Library, 2021. 2

[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 *European Conference on Computer Vision*, pages 405–421, 2020. 1, 2

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 7, 3

[33] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022. 7

[34] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. Practical svbrdf acquisition of 3d objects with unstructured flash photography. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018. 2

[35] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 2

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[37] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021. 1, 2

[38] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 2

[39] 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*, pages 195–206, 2007. 3

[40] Rui Xia, Yue Dong, Pieter Peers, and Xin Tong. Recovering shape and spatially-varying surface reflectance under unknown illumination. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 2

[41] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[42] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020. 2

[43] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 7, 3

[44] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021. 2

[45] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 2

[46] Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. Iron: Inverse rendering by optimizing neural sdfs and materials from photometric images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5565–5574, 2022. 2

[47] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[48] 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 Transactions on Graphics (ToG)*, 40(6):1–18, 2021. 1, 2, 6, 7

[49] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18643–18652, 2022. 2, 7

GS-IR: 3D Gaussian Splatting for Inverse Rendering

Supplementary Material

7. Implementation Details

We implement GS-IR in PyTorch framework [36] with CUDA extensions, and customized the baking-based method for GS-IR.

Representation. In the vanilla GS [24], each 3D Gaussian utilizes learnable $\mathcal{T} = \{\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{q}\}$ and $\mathcal{A} = \{\alpha, \boldsymbol{f}_c\}$ to describe its geometric properties and volumetric appearance respectively, where $\boldsymbol{p}$ denotes the position vector, $\boldsymbol{s}$ denotes the scaling vector, $\boldsymbol{q}$ denotes the unit quaternion for rotation, α denotes the opacity and $\boldsymbol{f}_c$ denotes spherical harmonics (SH) coefficients for view-dependent color. In GS-IR, we use $\boldsymbol{n}$ to present the normal vector of 3D Gaussian and extend the geometric properties as $\mathcal{T} = \{\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{q}, \boldsymbol{n}\}$. In addition, we introduce $\mathcal{M} = \{\boldsymbol{a}, \rho, m\}$ to describe the material of 3D Gaussian.

Training Details. We use the Adam optimizer [27] for training, and the training process includes the initial stage (*cf*. Sec. 4.1) and decomposition stage (*cf*. Sec. 4.3). In the initial stage, we minimize color reconstruction loss $\mathcal{L}_c$ and normal loss $\mathcal{L}_n$ (*cf*. Eq. (9)) to optimize $\mathcal{T}, \mathcal{A}$ for 30K iterations. In the decomposition stage, we fix $\mathcal{T}, \mathcal{A}$ and minimize the proposed decomposition loss $\mathcal{L}_d$ (*cf*. Eq. (16)) to merely optimize $\mathcal{M}$ for 10K iterations. The total optimization is running on a single V100 GPU.

Loss Definition. In the initial stage, the supervision loss $\mathcal{L}_{\text{init}}$ consists of the $L1$ color reconstruction loss $\mathcal{L}_c$ and our proposed normal loss $\mathcal{L}_n$:

$$\begin{aligned} \mathcal{L}_{\text{init}} &= \mathcal{L}_c + \mathcal{L}_n \\ \mathcal{L}_n &= \mathcal{L}_{n\text{-}p} + \lambda_{n\text{-}TV}\, TV_{\text{normal}} \end{aligned} \tag{16}$$

the smoothing term TV_{normal} in our proposed normal loss $\mathcal{L}_n$ is a total variation (TV) loss conditioned by the predicted normal map $\hat{\boldsymbol{N}}$ and the given reference image $\boldsymbol{I}$:

$$\begin{aligned} \triangle_{ij}^{\hat{\boldsymbol{N}}} &= \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i-1,j}|\right)(\hat{\boldsymbol{N}}_{i,j} - \hat{\boldsymbol{N}}_{i-1,j})^2 + \\ &\quad \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i,j-1}|\right)(\hat{\boldsymbol{N}}_{i,j} - \hat{\boldsymbol{N}}_{i,j-1})^2, \\ TV_{\text{normal}} &= \frac{1}{|\hat{\boldsymbol{N}}|}\sum_{i,j}\triangle_{ij}^{\hat{\boldsymbol{N}}}. \end{aligned} \tag{17}$$

In the decomposition stage, the supervision loss $\mathcal{L}_d$ includes $\mathcal{L}_{\text{shade}}, \mathcal{L}_{\text{material}}, \mathcal{L}_{\text{light}}$:

$$\mathcal{L}_d = \underbrace{\left\| \boldsymbol{I} - \hat{\boldsymbol{I}}^{\text{shade}}(\hat{\boldsymbol{M}}, \hat{\boldsymbol{E}}, \mathcal{V}^{\text{illu}}) \right\|}_{\mathcal{L}_{\text{shade}}} + \underbrace{\lambda_{\boldsymbol{M}}\ TV_{\text{mat}}}_{\mathcal{L}_{\text{material}}} + \underbrace{\lambda_{\boldsymbol{E}}\ TV_{\text{light}}}_{\mathcal{L}_{\text{light}}}, \tag{18}$$

the smoothing term TV_{mat} in Eq. (18) is a TV loss similar to TV_{normal} in Eq. (17):

$$\begin{aligned} \triangle_{ij}^{\hat{\boldsymbol{M}}} &= \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i-1,j}|\right)(\hat{\boldsymbol{M}}_{i,j} - \hat{\boldsymbol{M}}_{i-1,j})^2 + \\ &\quad \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i,j-1}|\right)(\hat{\boldsymbol{M}}_{i,j} - \hat{\boldsymbol{M}}_{i,j-1})^2, \\ TV_{\text{mat}} &= \frac{1}{|\hat{\boldsymbol{M}}|}\sum_{i,j}\triangle_{ij}^{\hat{\boldsymbol{M}}}, \end{aligned} \tag{19}$$

where $\hat{\boldsymbol{M}}$ is the predicted material map. Unlike the above two smoothing terms, TV_{light} is defined as:

$$\begin{aligned} \triangle_{ij}^{\hat{\boldsymbol{E}}} &= (\hat{\boldsymbol{E}}_{i,j} - \hat{\boldsymbol{E}}_{i-1,j})^2 + (\hat{\boldsymbol{E}}_{i,j} - \hat{\boldsymbol{E}}_{i,j-1})^2, \\ TV_{\text{light}} &= \frac{1}{|\hat{\boldsymbol{E}}|}\sum_{i,j}\triangle_{ij}^{\hat{\boldsymbol{E}}}. \end{aligned} \tag{20}$$

During training, we set $\lambda_{n\text{-}TV}, \lambda_{\boldsymbol{M}}, \lambda_{\boldsymbol{E}}$ to $5.0, 1.0, 0.01$. And we study the efficacy of these smoothing terms in Sec. 11.

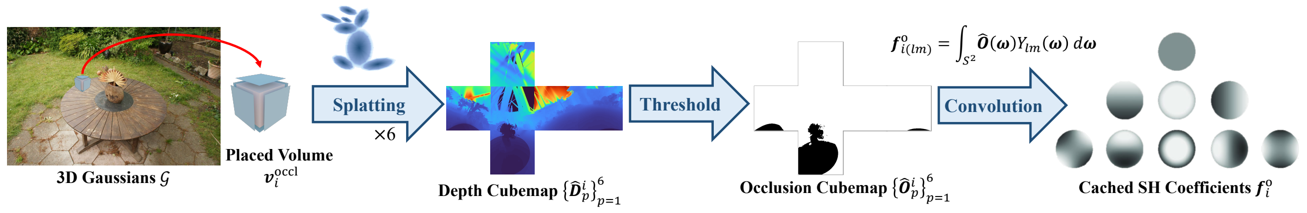
8. Occlusion Caching and Recovery

In the baking stage (*cf*. Sec. 4.2), we introduce SH architectures and cache occlusion into occlusion volumes $\mathcal{V}^{\text{occl}}$ as illustrated in Fig. 10a. For each volume $\boldsymbol{v}_i^{\text{occl}} \subset \mathcal{V}^{\text{occl}}$, we set six cameras with FoV of 90° and non-overlapping each other, and perform six render passes to obtain the depth cubemap $\{\hat{\boldsymbol{D}}_p^i\}_{p=1}^6$. Then we convert $\{\hat{\boldsymbol{D}}_p^i\}_{p=1}^6$ into the occlusion cubemap $\{\hat{\boldsymbol{O}}_p^i\}_{p=1}^6$ and store the principal components of occlusion into SH coefficients $\boldsymbol{f}_{\boldsymbol{x}}^o$.

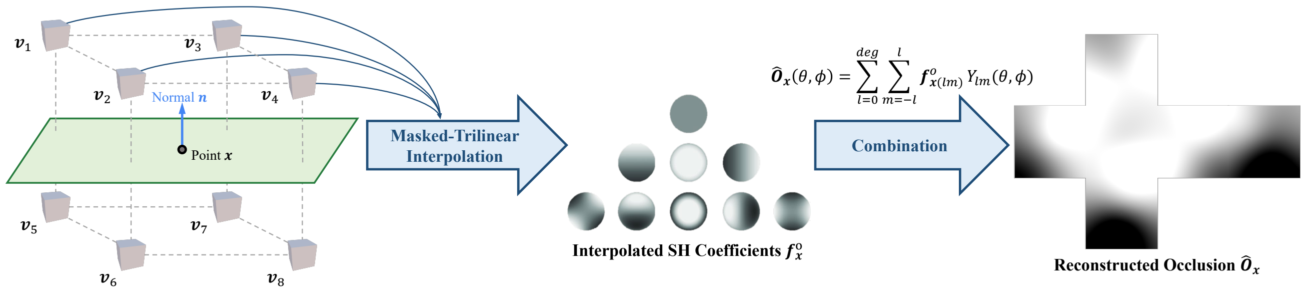
In the decomposition stage, we recover the ambient occlusion (AO) for each surface point $\boldsymbol{x}$ from occlusion volumes $\mathcal{V}^{\text{occl}}$. The first step is to get the coefficients $\boldsymbol{f}_{\boldsymbol{x}}^o$ of the point $\boldsymbol{x}$. Considering that AO of the point $\boldsymbol{x}$ only calculates the occlusion integral of the upper hemisphere Ω of the normal $\boldsymbol{n}$, we thus conduct masked-trilinear interpolation to get the correct coefficients. As illustrated in Fig. 10b, for the given point $\boldsymbol{x}$ with normal $\boldsymbol{n}$, we firstly find the eight nearest volumes $\{\boldsymbol{v}_k\}_{k=1}^8$. In this case, each volume has position vector $\boldsymbol{p}_k$ and SH coefficients $\boldsymbol{f}_k^o$. Given the trilinear interpolation weights $\{w_k\}_{k=1}^8$ defined in vanilla trilinear interpolation, we get the coefficients $\boldsymbol{f}_{\boldsymbol{x}}^o$:

$$\begin{aligned} \tilde{w}_k &= \begin{cases} 0, & (\boldsymbol{p}_k - \boldsymbol{x}) \cdot \boldsymbol{n} \le 0 \\ w_k, & (\boldsymbol{p}_k - \boldsymbol{x}) \cdot \boldsymbol{n} > 0 \end{cases}, \\ \hat{w}_k &= \frac{\tilde{w}_k}{\sum_{k=1}^8 \tilde{w}_k}, \\ \boldsymbol{f}_{\boldsymbol{x}(lm)}^{\text{o}} &= \sum_{k=1}^8 \hat{w}_k \boldsymbol{f}_{k(lm)}^{\text{o}}. \end{aligned} \tag{21}$$

The weights in trilinear interpolation satisfy $\sum_{k=1}^8 w_k = 1$


3D Gaussians $\mathcal{G}$
Placed Volume v_i^{occl}
Splatting
×6
Depth Cubemap $\{\hat{D}_p^i\}_{p=1}^6$
Threshold
Occlusion Cubemap $\{\hat{O}_p^i\}_{p=1}^6$
$f_{i(lm)}^o = \int_{S^2} \hat{O}(\omega) Y_{lm}(\omega)\, d\omega$
Convolution
Cached SH Coefficients f_i^o

(a) Occlusion caching from the pretrained 3D Gaussians $\mathcal{G}$ in the baking stage.


v_1
v_2
v_3
v_4
Normal n
Point x
v_5
v_6
v_7
v_8
Masked-Trilinear Interpolation
Interpolated SH Coefficients f_x^o
$\hat{O}_x(\theta,\phi) = \sum_{l=0}^{deg} \sum_{m=-l}^{l} f_{x(lm)}^o Y_{lm}(\theta,\phi)$
Combination
Reconstructed Occlusion $\hat{O}_x$

(b) Occlusion recovery from occlusion volumes $\mathcal{V}^{\text{occl}}$ in the decomposition stage.

Figure 10. Occlusion caching and recovery in GS-IR.

Scene	Method	Normal MAE ↓	Novel View Synthesis PSNR ↑	Novel View Synthesis SSIM ↑	Novel View Synthesis LPIPS ↓	Albedo PSNR ↑	Albedo SSIM ↑	Albedo LPIPS ↓	Relight PSNR ↑	Relight SSIM ↑	Relight LPIPS ↓
Lego	NeRFactor	9.767	26.076	0.881	0.151	25.444	0.937	0.112	23.246	0.865	0.156
	InvRender	9.980	24.391	0.883	0.151	21.435	0.882	0.160	20.117	0.832	0.171
	NVDiffrec	12.486	30.056	0.945	0.059	21.353	0.849	0.166	20.088	0.844	0.114
	TensoIR	5.980	34.700	0.968	0.037	25.240	0.900	0.145	28.581	0.944	0.081
	Ours	8.078	34.379	0.968	0.036	24.958	0.889	0.143	23.256	0.842	0.117
Hotdog	NeRFactor	5.579	24.498	0.940	0.141	24.654	0.950	0.142	22.713	0.914	0.159
	InvRender	3.708	31.832	0.952	0.089	27.028	0.950	0.094	27.630	0.928	0.089
	NVDiffrec	5.068	34.903	0.972	0.054	26.057	0.920	0.116	19.075	0.885	0.118
	TensoIR	4.050	36.820	0.976	0.045	30.370	0.947	0.093	27.927	0.933	0.115
	Ours	4.771	34.116	0.972	0.049	26.745	0.941	0.088	21.572	0.888	0.140
Armadillo	NeRFactor	3.467	26.479	0.947	0.095	28.001	0.946	0.096	26.887	0.944	0.102
	InvRender	1.723	31.116	0.968	0.057	35.573	0.959	0.076	27.814	0.949	0.069
	NVDiffrec	2.190	33.664	0.983	0.031	38.844	0.969	0.076	23.099	0.921	0.063
	TensoIR	1.950	39.050	0.986	0.039	34.360	0.989	0.059	34.504	0.975	0.045
	Ours	2.176	39.287	0.980	0.039	38.572	0.986	0.051	27.737	0.918	0.091
Ficus	NeRFactor	6.442	21.664	0.919	0.095	22.402	0.928	0.085	20.684	0.907	0.107
	InvRender	4.884	22.131	0.934	0.057	25.335	0.942	0.072	20.330	0.895	0.073
	NVDiffrec	4.567	22.131	0.946	0.064	30.443	0.894	0.101	17.260	0.865	0.073
	TensoIR	4.420	29.780	0.973	0.041	27.130	0.964	0.044	24.296	0.947	0.068
	Ours	4.762	33.551	0.976	0.031	30.867	0.948	0.053	24.932	0.893	0.081

Table 5. Per-scene results on TensoIR Synthetic dataset. For albedo reconstruction results, we follow NeRFactor [48] and scale each RGB channel by a global scalar.

After performing masked-trilinear interpolation, the occlusion $\hat{\boldsymbol{O}}_{\boldsymbol{x}}$ is written as:

$$\hat{\boldsymbol{O}}_{\boldsymbol{x}}(\theta,\phi) = \sum_{l=0}^{deg} \sum_{m=-l}^{l} \boldsymbol{f}_{\boldsymbol{x}(lm)}^{o} Y_{lm}(\theta,\phi). \tag{22}$$

For indirect illumination $I_{\text{d}}^{\text{indir}}$ in Eq. (13), we recover it from the volumes $\mathcal{V}^{\text{illu}}$ via vanilla trilinear interpolation.

Method	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
NeRF++	22.64	20.31	24.32	24.34	22.20	28.87	26.38	27.80	29.15
Plenoxels	21.91	20.10	23.49	20.66	22.25	27.59	23.62	23.42	24.67
INGP-Base	22.19	20.35	24.60	23.63	22.36	29.27	26.44	28.55	30.34
INGP-Big	22.17	20.65	25.07	23.47	22.37	29.69	26.69	29.48	30.69
Mip-NeRF 360	24.40	21.64	26.94	26.36	22.81	29.69	26.69	29.48	30.69
3DGS	25.25	21.52	27.41	26.55	22.49	30.63	28.70	30.32	31.98
Ours	23.80	20.57	25.72	25.37	21.79	28.79	26.22	27.99	28.18

Table 6. PSNR scores for Mip-NeRF360 scenes.

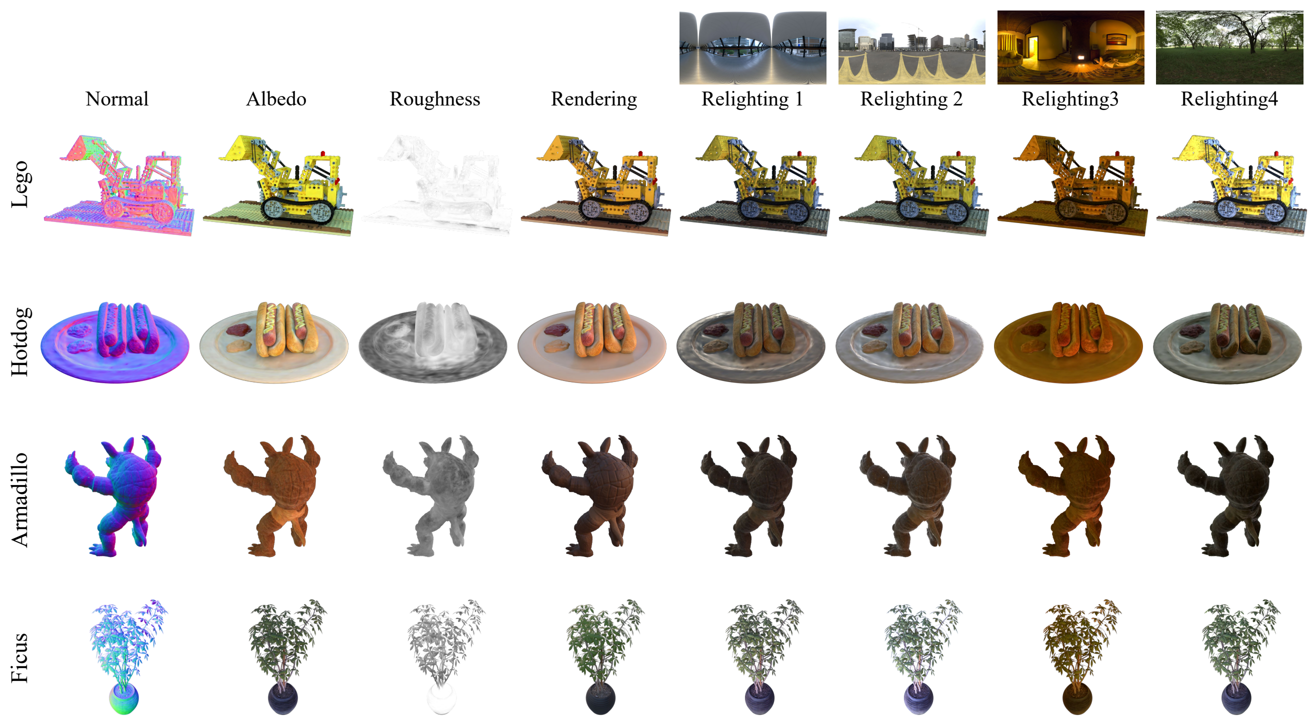


Figure 11. Visualization of our inverse rendering and relighting results on TensoIR Synthetic dataset.

Method	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
NeRF++	0.526	0.453	0.635	0.594	0.530	0.530	0.802	0.816	0.876
Plenoxels	0.496	0.431	0.606	0.523	0.509	0.842	0.759	0.648	0.814
INGP-Base	0.491	0.450	0.649	0.574	0.518	0.855	0.798	0.818	0.890
INGP-Big	0.512	0.486	0.701	0.594	0.542	0.871	0.817	0.858	0.906
Mip-NeRF 360	0.693	0.583	0.816	0.746	0.632	0.913	0.895	0.920	0.939
3DGS	0.771	0.605	0.868	0.775	0.638	0.914	0.905	0.922	0.938
Ours	0.706	0.543	0.804	0.716	0.586	0.867	0.839	0.867	0.883

Table 7. SSIM scores for Mip-NeRF360 scenes.

Method	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
NeRF++	0.455	0.466	0.331	0.416	0.466	0.335	0.351	0.260	0.291
Plenoxels	0.506	0.521	0.386	0.503	0.540	0.419	0.441	0.447	0.398
INGP-Base	0.487	0.481	0.312	0.450	0.489	0.301	0.342	0.254	0.227
INGP-Big	0.446	0.441	0.257	0.421	0.450	0.261	0.306	0.195	0.205
Mip-NeRF 360	0.289	0.345	0.164	0.254	0.338	0.211	0.203	0.126	0.177
3DGS	0.205	0.336	0.103	0.210	0.317	0.220	0.204	0.129	0.205
Ours	0.259	0.371	0.158	0.258	0.372	0.279	0.260	0.188	0.264

Table 8. LPIPS scores for Mip-NeRF360 scenes.

9. Results on TensoIR Synthetic Dataset

Tab. 5 provides the results on normal estimation, novel view synthesis, albedo reconstruction, and relighting for all four scenes. We also visualize the inverse rendering and relighting results of GS-IR in Fig. 11.

10. Results on Mip-NeRF 360

For Mip-NeRF 360 [5], a dataset captured from the real world, we list the results on novel view synthesis (*i.e.* PSNR, SSIM, and LPIPS) of GS-IR and some NeRF variants [17, 32, 43] in Tabs. 6 to 8. In addition, we provide the normal estimation, novel view synthesis, and relighting results of all seven publicly available scenes in Fig. 12.

11. Ablation on Loss

The loss in GS-IR consists of contrast terms and smoothing terms. For contrast terms, we set the weights of color reconstruction loss $\mathcal{L}_c$, normal penalty loss $\mathcal{L}_{n\text{-}p}$, and shade loss $\mathcal{L}_{\text{shade}}$ to 1, which is intuitive. And the smoothing terms include TV_{normal}, TV_{mat}, and TV_{light}, we evaluate their efficacy by adjusting their weights (*i.e.* $\lambda_{n\text{-}TV}$, $\lambda_{\boldsymbol{E}}$, and $\lambda_{\boldsymbol{M}}$), and the ablation results are shown in Tab. 9.

			Normal	Novel View Synthesis			Albedo		
$\lambda_{n\text{-}TV}$	$\lambda_{\boldsymbol{E}}$	$\lambda_{\boldsymbol{M}}$	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
			5.030	35.170	0.970	0.042	30.083	0.938	0.090
✓			**4.948**	35.330	**0.974**	0.039	30.216	0.940	0.088
✓	✓		**4.948**	35.230	0.972	0.040	30.236	0.940	0.087
✓		✓	**4.948**	35.314	0.973	**0.038**	30.275	**0.941**	0.085
✓	✓	✓	**4.948**	**35.333**	**0.974**	0.039	**30.286**	**0.941**	**0.084**

Table 9. **Analysis of the impact of different loss terms on the TensoIR dataset.** ✓indicates setting the smoothing term to be valid.

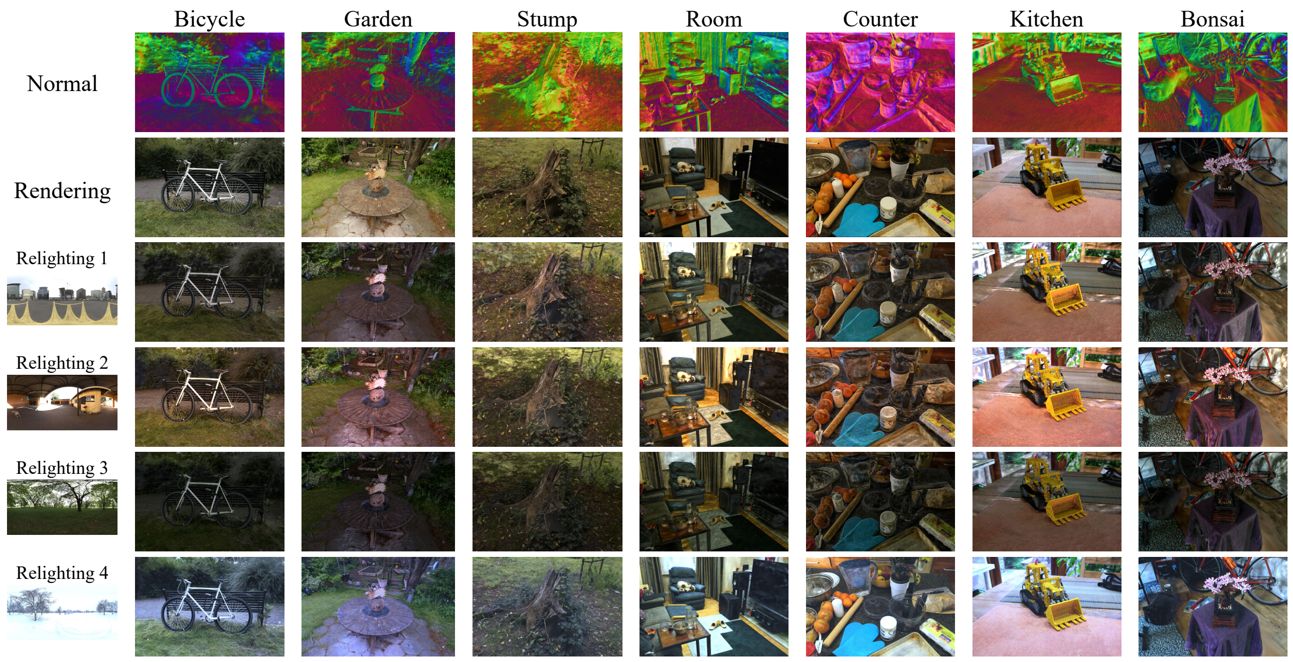


Figure 12. Visualization of our inverse rendering and relighting results on the Mip-NeRF 360 dataset.