GHPT: Real-Time Relightable Gaussian Splatting using Hybrid Path Tracing

Jinyang Bo* Fan Dou* Wenrui Quan Shangxun Liu Yang Xu† Yuhe Zhang
Kang Li Guohua Geng
College of Computer Science, Northwest University, China

Abstract

3D Gaussian splatting (3DGS) has emerged as a promising approach for high-fidelity 3D scene representation. However, relighting and composition of Gaussian splatting remain challenging because path tracing is not directly applicable. Existing relighting methods for Gaussian splatting typically adopt either approximate rendering formulations or rely on Gaussian ray tracing, yielding low relighting performance and low rendering efficiency. To address these limitations, we propose Gaussian hybrid path tracing (GHPT), a three-stage framework to acquire relightable Gaussian splatting models. The first stage utilizes planar-based Gaussian splatting reconstruction representation (PGSR) to enable multi-view consistent depth rendering and reconstruct the surface mesh of a scene. The second stage performs physically-based differentiable rendering on the obtained mesh to reconstruct the material maps and the environment map. The third stage utilizes factorized inverse path tracing (FIPT) on the G-buffer rendered by the PGSR, and visibility and indirect illumination are evaluated by hardware-accelerated ray tracing on the mesh with the material maps and the environment map reconstructed in the second stage. Experiments demonstrate that the relighting performance of GHPT outperforms the baselines, and our method can perform real-time relighting and composition of Gaussian splatting.

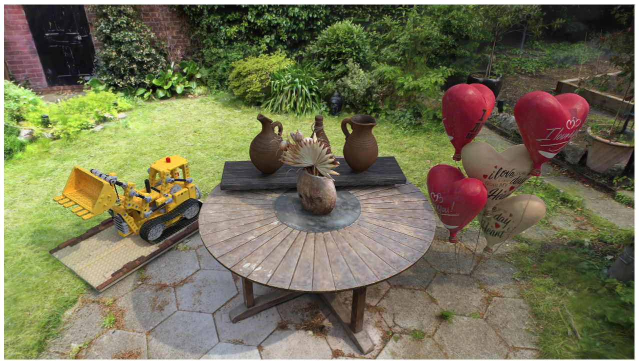
Figure 1. Real-time Gaussian splatting relighting and composition under the CITY environment map. 113 fps at 1920×1080, NVIDIA GeForce RTX 4080.

1. Introduction

In recent years, explicit volumetric radiance fields, such as 3D Gaussian splatting (3DGS) [20] have gained significant attention in the computer vision and computer graphics communities. However, as a novel view synthesis method, 3DGS cannot be used as an asset to build a complex scene as conventional mesh models because it cannot be relit, and inter-object effects such as shadows and interreflections cannot be simulated, which limits its applications in many fields such as AR/VR. Relighting and composition of Gaussian splatting is challenging because conventional photorealistic rendering techniques such as Monte Carlo path tracing cannot be directly employed on it. Although many attempts have been made to relight and composite Gaussian splatting, most of them leverage simplified rendering formulations such as split-sum approximation, ambient occlusion, and approximate indirect global illumination [8, 10, 22], resulting in reduced realism. Other methods that model the full global illumination rely on Gaussian ray tracing [12, 13, 35], which leads to low rendering efficiency.

In this work, we present *Gaussian hybrid path tracing* (GHPT) to acquire relightable Gaussian splatting. Hybrid path tracing, which shoots rays from the rasterized G-buffer instead of the cameras, is widely used in many real-time renderers and game engines. Inspired by hybrid path tracing, we propose a hybrid rendering model based on the G-buffer rendered by PGSR, which is a modified GS model that can render multi-view consistent depth. We perform path tracing on the G-buffer, and evaluate visibility and indirect illumination using the underlying meshes generated by PGSR and physically-based differentiable rendering, which can leverage hardware-accelerated ray tracing of modern GPUs. During training, FIPT is utilized to make the optimization of material properties more efficient and stable. Experimental results demonstrate that our method has best relighting per-

*Equally contributed
†Corresponding author: xuyang@nwu.edu.cn

formance among the inverse rendering baselines based on GS, and can acquire Gaussian splatting models that support real-time relighting and scene composition.

The main contributions of this work are listed below:

- A hybrid physically-based rendering model based on deferred shading on the G-buffer generated by Gaussian splatting and path tracing on the underlying mesh to evaluate visibility and indirect illumination.
- A three-stage inverse rendering framework that acquires geometries and environment maps before material decomposition, which enables utilization of FIPT to improve training efficiency.
- A real-time renderer based on Vulkan that supports relighting and composition of GS models.

2. Related Work

2.1. Novel View Synthesis

The task of novel view synthesis aims to leverage a set of images captured from known viewpoints to generate photorealistic images from arbitrary, previously unseen viewpoints. NeRF [25] represents a scene as a continuous volumetric radiance field, which is parameterized and optimized by a neural network (MLP), thereby enabling the synthesis of photorealistic images. Subsequent research in novel view synthesis has predominantly focused on accelerating rendering [27, 30], enhancing image quality [2, 3], integrating explicit structural representations [27]. 3DGS [20] decomposes a scene into a large number of Gaussian primitives, which provides novel avenues for research across multiple domains, including geometry reconstruction [7, 15, 40], inverse rendering [8, 10, 13, 22, 35], and robotic perception and navigation. However, 3DGS relies on rasterization to project Gaussian primitives into screen space, which impedes accurate simulation of lighting effects based on ray tracing, such as soft shadows and indirect illumination. To overcome these limitations, 3DGRT [26] replaces rasterization with ray tracing, builds a bounding volume hierarchy (BVH), and leverages ray–triangle intersections to efficiently compute the accumulated radiance of primitives, but the rendering speed is significantly reduced.

2.2. Inverse Rendering

The aim of inverse rendering is to infer scene geometry, material properties, and scene illumination from images. Inverse rendering is extremely challenging due to the high-dimensional nonlinear interactions between light and surfaces and the interdependence of the decomposed material properties. Controlled-illumination [4, 5, 11, 24, 29, 41] approaches simplify the problem under restrictive lighting setups. Due to that physically-based differentiable rendering [16, 21, 23, 42] can faithfully model global illumination via path tracing for differentiable light transport, many inverse rendering methods based on explicit geometry representations are proposed [9, 14, 28, 34, 38, 43]. More recently, neural implicit-field methods [17, 34, 44, 45] leverage NeRF-style ray marching with learnable material and lighting parameters represented by spherical harmonics, spherical Gaussians, or environment maps for end-to-end optimization, but suffer from limited expressiveness and high ray-marching costs. To accelerate rendering, Gaussian splatting has been adapted for inverse rendering, which can be classified into forward and deferred rendering methods. Forward rendering methods [12, 35] shade Gaussians individually and then blend the shading results via alpha-blending, and deferred rendering methods [8, 10, 13, 22] rasterize G-buffer and then employ the rendering formulation based on the split-sum approximation or Gaussian ray tracing [26] on the G-buffer. Our method is a deferred rendering method maintaining both quality and efficiency because we perform hybrid path tracing on the G-buffer with the underlying meshes to render shadows and indirect illumination.

3. Preliminary

3D Gaussian Splatting 3DGS represents a scene explicitly as a set of Gaussian distributions G_i, each approximating a local neighborhood around points in a 3D point cloud. Each Gaussian distribution is defined as:

$$G_i(\boldsymbol{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \exp\Big(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\Big), \tag{1}$$

where $\boldsymbol{\mu}_i \in R^3$ denotes the center coordinates of point $\boldsymbol{p}_i \in \boldsymbol{P}$ and $\boldsymbol{\Sigma}_i \in R^{3\times3}$ is its covariance matrix. The covariance matrix $\boldsymbol{\Sigma}_i$ is factorized as:

$$\boldsymbol{\Sigma}_i = \boldsymbol{R}_i \boldsymbol{S}_i \boldsymbol{S}_i^\top \boldsymbol{R}_i^\top, \tag{2}$$

where $\boldsymbol{S}_i = \{s_x, s_y, s_z\}$ represents the scaling factors and $\boldsymbol{R}_i$ is a rotation matrix constructed from a unit quaternion q. 3DGS facilitates fast alpha-bending for rendering. Given a transformation matrix $\boldsymbol{W}$ and an intrinsic matrix $\boldsymbol{K}$, the Gaussian parameters transform from world to camera coordinates and project onto the 2D image plane as follows:

$$\boldsymbol{\mu}_i' = \boldsymbol{K}\boldsymbol{W}[\boldsymbol{\mu}_i, 1]^\top, \quad \boldsymbol{\Sigma}_i' = \boldsymbol{J}\boldsymbol{W}\boldsymbol{\Sigma}_i\boldsymbol{W}^\top\boldsymbol{J}^\top, \tag{3}$$

where $\boldsymbol{J}$ is the Jacobian matrix of the perspective projection approximation. Pixel color $C \in R^3$ is obtained through alpha-blending:

$$C = \sum_{i\in N} T_i \alpha_i c_i, \quad T_i = \prod_{j=1}^{i-1}(1-\alpha_j), \tag{4}$$

where α_i is derived from evaluating $G_i(u \mid \boldsymbol{\mu}_i', \boldsymbol{\Sigma}_i')$ scaled by a learned opacity value. The color $c_i \in R^3$ at Gaussian G_i depends on the viewing direction and is encoded by spherical harmonics (SH), The cumulative opacity T_i

accounts for occlusion among Gaussians along a viewing ray. The Gaussian primitive center $\boldsymbol{\mu}_i$ projects into camera coordinates as:

$$[x_i, y_i, z_i, 1]^\top = \boldsymbol{W}[\boldsymbol{\mu}_i, 1]^\top. \tag{5}$$

4. Method

4.1. Overview

Figure 2 outlines our proposed GS-based inverse rendering pipeline, which employs hybrid path tracing to evaluate both direct and indirect illumination. We employ a three-stage process: initially, we use PGSR to reconstruct mesh models from multi-view images. In the second stage, we leverage physically-based differentiable rendering to produce material maps for the mesh models and the environment maps, thereby providing the prerequisites for visibility and indirect illumination computation for FIPT. In the third stage, we employ FIPT on the G-buffer rendered by PGSR to generate pre-baked shadings and integrate this with GS-based inverse rendering to recover material properties for each Gaussian.

4.2. Multi-View Consistent Depth Rendering

The main purpose of this stage is to obtain multi-view consistent depth maps and the corresponding mesh model for the second and third stages.

Planar-Based Gaussian Splatting for Surface Reconstruction To achieve multi-view geometric consistency, we leverage PGSR [7], which compresses 3D Gaussians into 2D planar representations, and render plane distance and normal maps to obtain unbiased depth maps.

In PGSR, the normal map under the current viewpoint is obtained through alpha-blending:

$$\boldsymbol{n} = \sum_{i \in N} T_i \alpha_i \boldsymbol{R}_\mathrm{c}^\top \boldsymbol{n}_i, \tag{6}$$

where $\boldsymbol{R}_c$ is the rotation from the camera to the global world. The distance from the plane to the camera center can be expressed as:

$$d_i = \left(\boldsymbol{R}_\mathrm{c}^\top (\boldsymbol{\mu}_i - \boldsymbol{T}_\mathrm{c})\right)^\top (\boldsymbol{R}_\mathrm{c}^\top \boldsymbol{n}_i), \tag{7}$$

where $\boldsymbol{T}_\mathrm{c}$ is the camera center in the world. $\boldsymbol{\mu}_i$ is the center of Gaussian G_i. The final distance map under the current viewpoint is obtained through alpha-blending:

$$\mathcal{D} = \sum_{i \in N} T_i \alpha_i d_i. \tag{8}$$

After obtaining the distance and normal of the plane through alpha-blending, the depth map can be acquired by intersecting rays with the plane:

$$D(\boldsymbol{p}) = \frac{\mathcal{D}}{\boldsymbol{n}(\boldsymbol{p})\boldsymbol{K}^{-1}\tilde{\boldsymbol{p}}}, \tag{9}$$

where $\boldsymbol{p} = [u, v]^T$ denotes the 2D position on the image plane, $\tilde{\boldsymbol{p}}$ is the homogeneous coordinate of $\boldsymbol{p}$, and $\boldsymbol{K}$ is the intrinsic matrix of the camera.

Unlike other methods that render the depth map based on alpha-blending of the depth of Gaussians, which introduces inconsistency with the flat Gaussian shape and causes geometric conflicts, PGSR renders the normal and distance maps of the plane first and then converts them into the depth map, which ensures that the depth lies on the Gaussian flat plane. PGSR also introduces single-view geometric, multi-view photometric, and geometric consistency loss to ensure global geometry consistency.

Additionally, we incorporate a pretrained monocular surface-normal prior from StableNormal [39] as an additional supervision to counteract the influence of specular reflections. Concretely, we define a normal consistency loss:

$$\mathcal{L}_\mathrm{normal} = \sum (1 - \boldsymbol{n}_\mathrm{render} \cdot \boldsymbol{n}_\mathrm{mono}), \tag{10}$$

where $\boldsymbol{n}_\mathrm{mono}$ is the normal predicted by StableNormal and $\boldsymbol{n}_\mathrm{render}$ is the corresponding rendered normal in PGSR via alpha-blending. Minimizing $\mathcal{L}_\mathrm{normal}$ aligns the rendered normals with the monocular normal prior. Additionally, we introduce a binary cross-entropy loss [37] as other methods [12, 13] do to explicitly constrain the geometric structure:

$$\mathcal{L}_\mathrm{mask} = -\mathcal{M} \log \mathcal{O} - (1 - \mathcal{M}) \log(1 - \mathcal{O}), \tag{11}$$

where $\mathcal{M}$ denotes the object mask and $\mathcal{O}$ denotes the accumulated opacity.

4.3. Physically-Based Differentiable Rendering

The goal of this stage is to combine the mesh model obtained from the first stage with multi-view images to obtain a mesh model with material maps and an environment map.

The outgoing radiance at the surface point $\boldsymbol{x}$ can be evaluated by the rendering equation [18], which is an integral over the upper hemisphere Ω^+ if we ignore the emission term:

$$L_\mathrm{o}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{o}) = \int_{\Omega^+} L_\mathrm{i}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{i}) f_\mathrm{r}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})(\boldsymbol{n} \cdot \boldsymbol{\omega}_\mathrm{i}) \mathrm{d}\boldsymbol{\omega}_\mathrm{i}, \tag{12}$$

where $L_\mathrm{o}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{o})$ denotes the outgoing radiance at $\boldsymbol{x}$ from direction $\boldsymbol{\omega}_\mathrm{o}$, $L_\mathrm{i}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{i})$ denotes the incident radiance arriving from direction $\boldsymbol{\omega}_\mathrm{i}$, $f_\mathrm{r}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})$ is the bidirectional reflectance distribution function (BRDF) at $\boldsymbol{x}$ from $\boldsymbol{\omega}_\mathrm{i}$ to $\boldsymbol{\omega}_\mathrm{o}$, and $\boldsymbol{n}$ is the surface normal at $\boldsymbol{x}$. Path tracing can be used to solve the rendering equation, in which the hemisphere integral in Eq. (12) is approximated by drawing N samples $\{\boldsymbol{\omega}_\mathrm{i}\}$ from a particular distribution with the probability density function (PDF) $p(\boldsymbol{\omega}_\mathrm{i})$. If we assume the scene is illuminated by an environment map with incident radiance $L_\mathrm{i}^\mathrm{env}(\boldsymbol{\omega}_i)$, we can divide the outgoing radiance $L_\mathrm{o}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{o})$

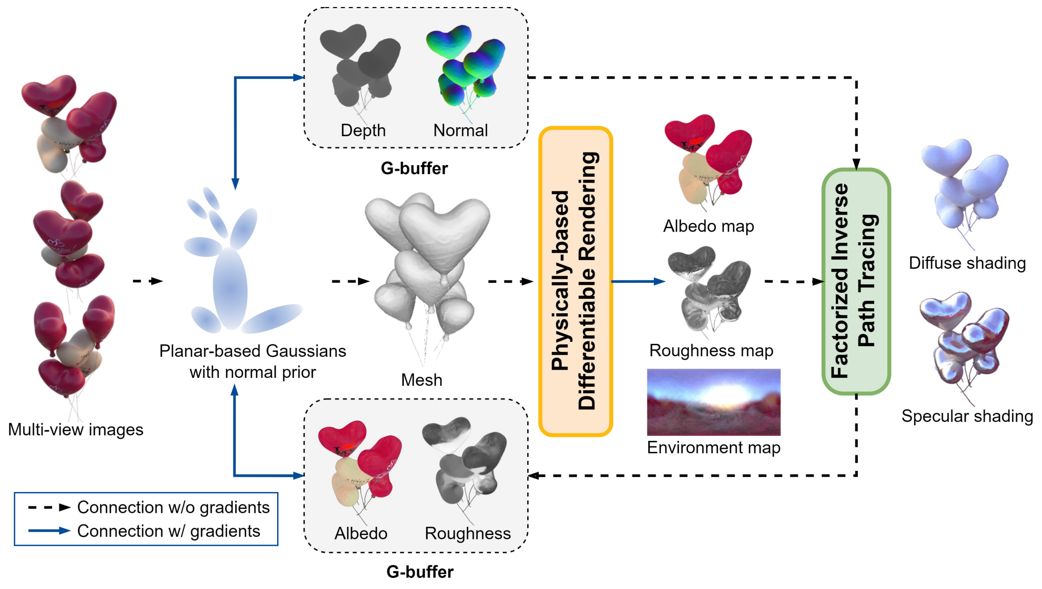

Figure 2. Our three-stage GS-based inverse rendering pipeline using hybrid path tracing. In the first stage, the scene geometry is reconstructed from multi-view images through planar-based Gaussians guided by normal priors. The second stage employs physically-based differentiable rendering to recover material properties of the mesh and the environment map. In the final stage, we use FIPT to recover material property of each Gaussian and take into account visibility and indirect illumination by ray tracing the underlying mesh.

into direct and indirect components, where the direct component $L_{\mathrm{o}}^{\mathrm{direct}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}})$ can be estimated by

$$L_{\mathrm{o}}^{\mathrm{direct}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) = \int_{\Omega^+} L_{\mathrm{i}}^{\mathrm{env}}(\boldsymbol{\omega}_{\mathrm{i}}) f_{\mathrm{r}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}})(\boldsymbol{n}\cdot\boldsymbol{\omega}_{\mathrm{i}})\mathrm{d}\boldsymbol{\omega}_{\mathrm{i}} \approx \frac{1}{N}\sum_{k=1}^{N} \frac{L_{\mathrm{i}}^{\mathrm{env}}(\boldsymbol{\omega}_{\mathrm{i}}^k) f_{\mathrm{r}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}}^k, \boldsymbol{\omega}_{\mathrm{o}})(\boldsymbol{n}\cdot\boldsymbol{\omega}_{\mathrm{i}}^k)}{p(\boldsymbol{\omega}_{\mathrm{i}}^k)}, \tag{13}$$

and the indirect component $L_{\mathrm{o}}^{\mathrm{ind}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}})$ can be estimated by tracing rays recursively as:

$$L_{\mathrm{o}}^{\mathrm{ind}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) = \int_{\Omega^+} L_{\mathrm{i}}^{\mathrm{ind}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}})\, f_{\mathrm{r}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}})\,(\boldsymbol{n}\cdot\boldsymbol{\omega}_{\mathrm{i}})\,\mathrm{d}\boldsymbol{\omega}_{\mathrm{i}} \approx \frac{1}{N}\sum_{k=1}^{N} \frac{L_{\mathrm{i}}^{\mathrm{ind}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}}^k)\, f_{\mathrm{r}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}}^k, \boldsymbol{\omega}_{\mathrm{o}})\,(\boldsymbol{n}\cdot\boldsymbol{\omega}_{\mathrm{i}}^k)}{p(\boldsymbol{\omega}_{\mathrm{i}}^k)}. \tag{14}$$

Our physically-based differentiable renderer has two passes for each Monte Carlo sample. The first pass is not differentiable, in which we obtain the primary hit, the visibility of samples to the environment map, and the indirect outgoing radiance, and store them for the second pass. In the second pass, we use the visibility, the environment map, and the BRDF at the primary hit to evaluate the direct radiance and then add the indirect radiance to acquire the final radiance. After the contributions of all samples are accumulated, we compute the per-pixel L1 loss between the final radiance and the corresponding value in the target image to optimize the BRDF and the environment map through backward propagation of the gradients. Additionally, we leverage next event estimation (NEE) to directly sample the environment map based on an alias table, and combine BRDF importance sampling and environment map importance sampling using multiple importance sampling (MIS) with the balance heuristic [36]. Please refer to the supplementary document for more details.

4.4. Factorized Inverse Path Tracing on Gaussian Splatting

The aim of this stage is to decompose the material properties of a scene represented by Gaussians based on the environment map and the mesh with material maps optimized by the previous stages. The textured mesh is used to evaluate visibility and indirect illumination. As previous methods do, we assign each Gaussian diffuse additional material properties: albedo a and roughness r, and the predicted albedo map $\mathcal{A}$ and roughness map $\mathcal{R}$ can be obtained through alpha-blending:

$$\{\mathcal{A}, \mathcal{R}\} = \sum_{i\in N} T_i \alpha_i \{a_i, r_i\}. \tag{15}$$

Since the environment map remains unchanged and we have the mesh with material maps, we can adopt FIPT [38] to improve the training efficiency. FIPT factorizes the light transport integral by pre-baking diffuse and specular shading maps, thereby separating the material properties out from the rendering integral. In FIPT, we can rewrite Eq. (12) as:

$$L_\mathrm{o}(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{o}) = k_\mathrm{d}L_\mathrm{d}(\boldsymbol{x}) + k_\mathrm{s}L_\mathrm{s}^0(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{o},r) + L_\mathrm{s}^1(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{o},r), \tag{16}$$

$$\begin{aligned} L_\mathrm{d}(\boldsymbol{x}) &= \int_{\Omega+} L_\mathrm{i}(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{i})\frac{\boldsymbol{n}\cdot\boldsymbol{\omega}_\mathrm{i}}{\pi}d\boldsymbol{\omega}_\mathrm{i}, \\ L_\mathrm{s}^0(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{o},\sigma) &= \int_{\Omega+} L_\mathrm{i}(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{i})\frac{F_0DG}{4(\boldsymbol{n}\cdot\boldsymbol{\omega}_\mathrm{o})}d\boldsymbol{\omega}_\mathrm{i}, \\ L_\mathrm{s}^1(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{o},\sigma) &= \int_{\Omega+} L_\mathrm{i}(\boldsymbol{x},\boldsymbol{\omega}_\mathrm{i})\frac{F_1DG}{4(\boldsymbol{n}\cdot\boldsymbol{\omega}_\mathrm{o})}d\boldsymbol{\omega}_\mathrm{i}, \end{aligned} \tag{17}$$

where $k_\mathrm{d} = a$ and $k_\mathrm{s} = 0.04$ are the diffuse and specular reflectances, r denotes the roughness, L_d is the diffuse shading, and L_s^0 and L_s^1 are the specular shadings corresponding to the two Fresnel components $F_0 = 1-(1-\boldsymbol{h}\cdot\boldsymbol{\omega}_\mathrm{i})^5$ and $F_1 = (1-\boldsymbol{h}\cdot\boldsymbol{\omega}_\mathrm{i})^5$. The specular shadings can be further approximated by linearly interpolating the shadings pre-computed at various roughness levels $L_\mathrm{s}^*(\cdot,r) \approx \mathrm{lerp}(\{L_\mathrm{s}^*(\cdot,r_i)\}_{i=1}^6, r)$, where $\{r_i\}_{i=1}^6$ is uniformly sampled between $(0,1)$. With the factorization, k_d, k_s, and r are all separated out of the integral, and the shadings L_d, L_s^0, and L_s^1 can be precomputed for each view with a large number of samples per pixel (spp) for variance reduction, and then be queried during training to accelerate the training process.

Although PGSR can ensure multi-view geometric consistency, the surface of the reconstructed mesh still slightly differs from the rendered G-buffer, which may cause self-intersection in hybrid ray tracing. To alleviate this issue, we ignore the intersections with a hit distance less than a small threshold (we use 0.1 in the experiments) and on the back face. Please refer to the supplementary document for more details.

After generating the diffuse and specular shadings, we conduct material decomposition for the Gaussians to disentangle reflectance from shading.

Loss Function We employ the following loss function to optimize the albedo and roughness of each Gaussian:

$$\mathcal{L} = \mathcal{L}_\mathrm{shade} + \lambda_\mathrm{a}\mathcal{L}_\mathrm{s,a} + \lambda_\mathrm{r}\mathcal{L}_\mathrm{s,r}, \tag{18}$$

where $\mathcal{L}_\mathrm{shade}$ denotes the L1 loss between the target image and the rendered image obtained using FIPT with the predicted current albedo and roughness maps. $\mathcal{L}_\mathrm{s,a}$ and $\mathcal{L}_\mathrm{s,r}$ are the edge-aware smoothness losses of predicted albedo and roughness maps to regularize them. Taking the albedo map as an example, we define the smoothness loss of the predicted albedo map as:

$$\mathcal{L}_\mathrm{s,a} = \|\nabla\mathcal{A}\|\exp(-\|\nabla I_\mathrm{gt}\|), \tag{19}$$

where $\mathcal{A}$ and $\mathcal{R}$ are computed by Eq. (15), and I_gt is the target image.

5. Experiments

5.1. Datasets and Metrics

The performance of our method is validated by two datasets: SYNTHETIC4RELIGHT [45] and TENSOIR SYNTHETIC [17]. For quantitative comparisons, we adhere to prior work and utilize a suite of evaluation metrics to assess our method. Specifically, we employ PSNR, SSIM, and LPIPS metrics to quantitatively evaluate image quality across novel view synthesis, albedo, and relighting. In the real-time relighting and composition experiments, we use GARDEN, KITCHEN, and ROOM from the MIP-NERF 360 dataset [2] as the scenes, and the trained models from SYNTHETIC4RELIGHT [45] and TENSOIR SYNTHETIC datasets are inserted into the three scenes.

5.2. Training

We use a computer equipped with an Intel Core i7-13700K CPU with 32 GB RAM and an NVIDIA GeForce RTX 4080 GPU in our experiments.

PGSR Training and Mesh Reconstruction During PGSR training, we set the weight of the normal prior loss $\mathcal{L}_\mathrm{normal}$ to 0.15 and the weight of the binary cross-entropy loss $\mathcal{L}_\mathrm{O}$ to 0.05. Then we reconstruct the surface meshes with a voxel size of 0.002 for the scenes in SYNTHETIC4RELIGHT [45] and TENSOIR SYNTHETIC [17] datasets, and 0.01 for the scenes in MIP-NERF 360 dataset. The reconstructed meshes are decimated to 500 k triangles using quadric error metrics. UV atlases of the meshes are generated by Blender [6] because UV coordinates are required by the differentiable renderer in the second stage to optimize the material maps.

Physically-Based Differentiable Rendering In this stage, we implement the physically-based differentiable renderer using SlangPy [19] because it supports hardware-accelerated ray tracing via D3D12/Vulkan and automatic differentiation via SLANG.D [1]. We optimize the material maps and the environment map of a scene. We utilize 256 spp, where 128 samples are for BRDF sampling and the other 128 samples are for environment map sampling. The resolutions of the albedo map, the roughness map, and the environment map are set to 4090×4096, 2048×2048, and 512×256, respectively. We set the initial values of the albedo map, the roughness map, and the environment map to 0, 0.8, and 0.5, respectively. We divide the training process into two phases, where the first phase is mainly for the environment map and the second one is for the material maps. The first phase takes 5000 iterations, and the Adam optimizer with an initial learning rate of 0.1 is applied for the material maps, and the initial learning rate for the environment map is set to 0.005. As NVDIFFREC [28] and NVDIFFRECMC [14] do, the gradients of the material parameters are divided by 8 at each iteration. We find that the above training process tends to bake the lighting in the albedo map instead of obtaining a high-contrast environment map. To better disentangle material and lighting, we enhance the contrast of the optimized environment map by raising it to the power of 1.35. In the second phase, we keep the environment map unchanged, reset the roughness map to the initial value, and continue to train the material maps for 1000 iterations. The initial learning rate of the material maps is set to 0.005.

FIPT on Gaussian Splatting We also use SlangPy to bake the diffuse and specular shadings, and we use PyTorch to perform material decomposition in this stage. We use 256 spp in training because FIPT only needs to generate diffuse and specular shadings once before iterative optimization. During inference, 256 spp are used, and spatial denoising is applied to the rendering results to further reduce the noise, and the window size of the edge-aware spatial filter is set to 9. The initial values of the albedo and the roughness are set to 0.5 and 0.8, respectively. The learning rate of albedo and roughness is set to 0.01, and the training iterations are set to 5000, and we set $\lambda_{\mathrm{s,a}}$ and $\lambda_{\mathrm{s,r}}$ in Eq. (18) to 0.025 and 0.05, respectively.

For SYNTHETIC4RELIGHT and TENSOIR SYNTHETIC datasets, the training process takes ∼40 minutes for geometry reconstruction, ∼15 minutes for physically-based differentiable rendering, and ∼5 minutes for FIPT on Gaussian splatting. For the MIP-NERF 360 dataset, the training process takes ∼2 hours, 40∼100 minutes depending on the scene complexity, and 10∼30 minutes for the three stages.

5.3. Comparison

We report error metrics and compare them to the NeRF/GS-based inverse rendering baselines. To deal with the inherent ambiguity between the albedo and the environment map, we follow the prior work [8, 13, 22, 35] to scale each RGB channel of the albedo map by a global scalar based on the ground truth. We use the official codes of R3DG [12] and IRGS [13] to generate the results. The quantitative comparison in Tab. 2 demonstrates that our method has the best relighting performance on SYNTHETIC4RELIGHT dataset, and also outperforms all baselines on TENSOIR SYNTHETIC dataset in terms of PSNR. Additionally, the SSIM scores of relighting are also comparable to the best. The PSNR scores of novel view synthesis on SYNTHETIC4RELIGHT dataset and albedo on TENSOIR SYNTHETIC dataset are also the best. We also provide the qualitative comparisons on the two datasets in Fig. 3 and Fig. 4. Please see the supplementary document for more qualitative comparisons.

5.4. Real-Time Relighting and Composition

To use the trained GS models for relighting and scene composition, we implement GHPT in a real-time renderer using the Vulkan Graphics API with the ray query extension to support hardware-accelerated ray tracing.

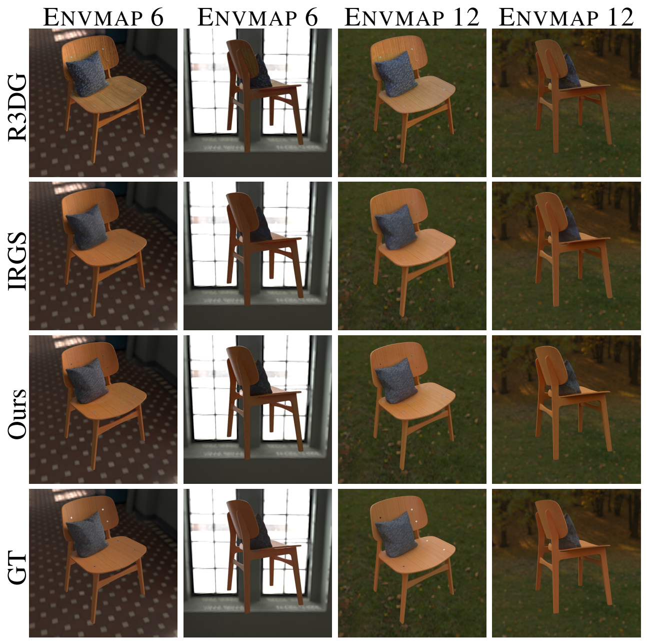

Figure 3. Qualitative comparison of relighting results under different environment maps on SYNTHETIC4RELIGHT dataset.

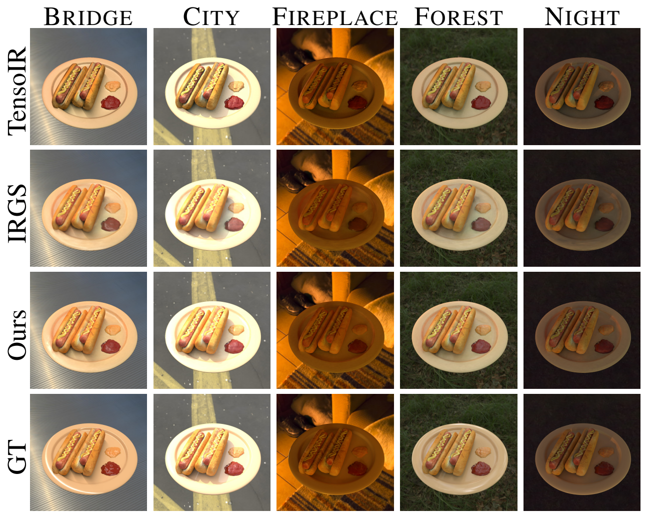

Figure 4. Qualitative comparison of relighting results under different environment maps on TENSOIR SYNTHETIC dataset.

G-Buffer Rendering We render the normal map and the depth map as the G-buffer using the tile-based software rasterizer instead of the graphics pipeline because normals and distances are required to be alpha-blended in PGSR, which requires high GPU memory bandwidth and is time-consuming for the graphics pipeline. We use the *Chained Scan Decoupled Lookback Decoupled Fallback* [33] and the

Table 1. Quantitative comparison on SYNTHETIC4RELIGHT dataset.

	Novel View Synthesis			Albedo			Relighting		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRFactor [44]	22.80	0.916	0.150	19.49	0.864	0.206	21.54	0.875	0.171
InvRender [45]	30.74	0.953	0.086	28.28	0.935	0.072	28.67	0.950	0.091
TensorIR [17]	35.80	0.978	0.049	29.69	0.951	0.079	30.58	0.946	0.065
GS-IR [22]	33.95	0.965	0.057	19.48	0.896	0.117	25.40	0.924	0.083
R3DG [12]	36.06	0.980	0.031	28.67	0.953	0.063	32.89	0.968	0.051
GI-GS [8]	35.42	0.973	0.042	24.68	0.931	0.085	27.36	0.945	0.070
IRGS [13]	35.66	0.970	0.050	33.07	0.953	0.060	34.76	0.963	0.055
Ours	37.11	0.976	0.037	32.20	0.950	0.074	35.87	0.971	0.045

Table 2. Quantitative comparison on TENSOIR SYNTHETIC dataset.

	Novel View Synthesis			Albedo			Relighting		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRFactor [44]	24.68	0.922	0.121	25.13	0.940	0.109	23.38	0.908	0.131
InvRender [45]	27.37	0.934	0.089	27.34	0.933	0.101	23.97	0.901	0.101
TensorIR [17]	35.09	0.976	0.041	29.28	0.950	0.085	28.83	0.950	0.077
GS-IR [22]	35.02	0.964	0.043	32.25	0.943	0.085	24.69	0.886	0.099
R3DG [12]	33.15	0.960	0.040	28.55	0.922	0.087	27.60	0.921	0.081
GI-GS [8]	36.75	0.973	0.037	31.97	0.941	0.086	24.70	0.886	0.106
IRGS [13]	35.58	0.965	0.048	33.40	0.954	0.076	30.63	0.935	0.075
SVG-IR [35]	36.71	0.975	0.033	30.34	0.951	0.074	31.10	0.946	0.056
Ours	35.57	0.967	0.038	33.67	0.938	0.088	32.46	0.948	0.057

Device Radix Sort algorithms developed by Smith [31, 32] to perform inclusive sum and radix sort.

Hybrid Path Tracing To achieve real-time performance, we use only 2 spp in hybrid path tracing, where one sample is for BRDF sampling and the other for environment map sampling. In addition to spatial filtering, temporal accumulation with a history length of 20 is applied to further reduce the noise. The rendering resolution is 1920×1080.

Figure 5 gives the rendering results without path tracing, with our hybrid path tracing, and the results of our method relit by two environment maps, BRIDGE and FIREPLACE. Since the rendering results without path tracing do not take into account visibility and indirect illumination, shadows and interreflections cannot be rendered, which leads to low realism. Our method can simulate a full global illumination solution and capture all inter-object effects, such as soft shadows, specular shadows, color bleeding, and glossy interreflections, which significantly enhances the realism of scene composition.

The timing breakdown of our real-time relighting and composition in these scenes is provided in Tab. 3. All timings are averaged over 100 frames. From the timings, we can see that the ROOM scene takes more time in path tracing than other scenes because the environment map is partially blocked by the walls, which increases the path length in path tracing. Please see the supplementary video for live demos.

Table 3. Timing breakdown in ms of the real-time relighting and composition

	GARDEN	KITCHEN	ROOM
G-buffer rendering	3.51	3.03	3.35
Hybrid path tracing	2.70	2.54	4.65
Denoising	1.74	1.75	1.75
Total	7.95	7.32	9.75

5.5. Ablation Study

We conduct ablation studies on SYNTHETIC4RELIGHT and TENSOIR SYNTHETIC datasets to investigate the influences of different components of our method on the relighting performance. The PSNR, SSIM, and LPIPS are reported in

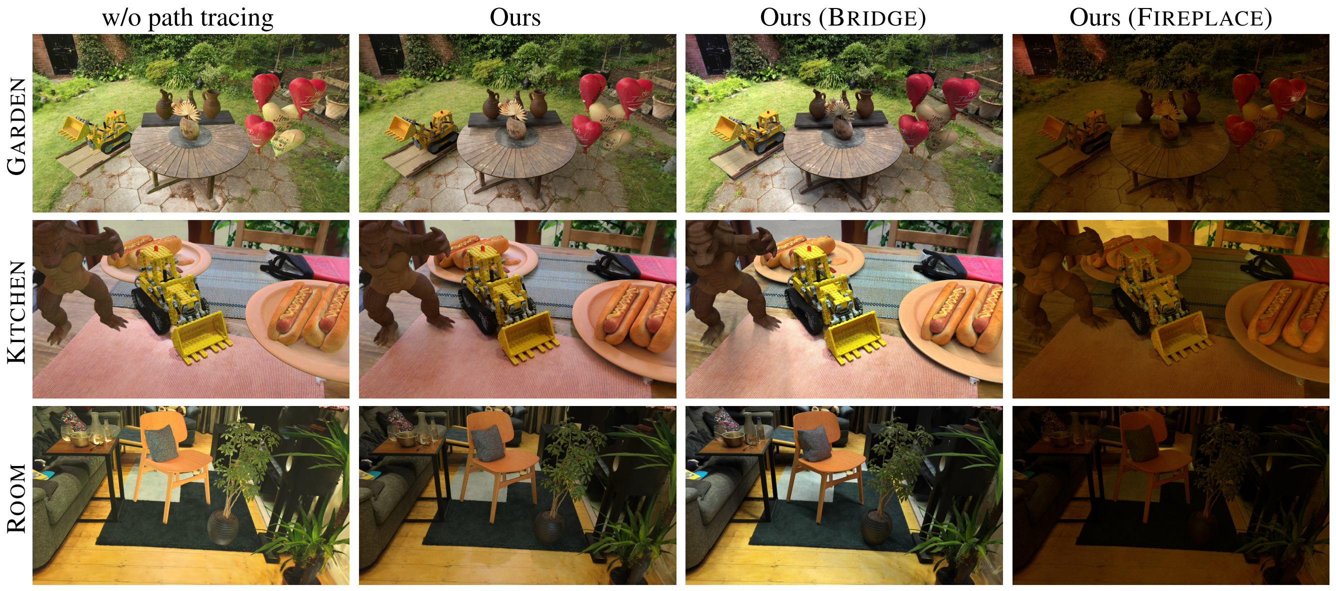

Figure 5. Real-time relighting and composition results in the GARDEN, KITCHEN, and ROOM scenes. From left to right: results without path tracing, results of our method, and results of our method relit by two environment maps, BRIDGE and FIREPLACE. Our method can produce more realistic results due to the presence of shadows and indirect illumination.

Table 4. Ablation studies on SYNTHETIC4RELIGHT and TENSOIR SYNTHETIC datasets on the relighting performance.

	SYNTHETIC4RELIGHT [45]			TENSOIR SYNTHETIC [17]		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Underlying mesh	34.83	0.964	**0.044**	31.81	0.940	0.062
w/o denoising	35.46	0.952	0.062	32.06	0.929	0.084
16 spp (relight)	34.87	0.954	0.066	31.69	0.931	0.080
64 spp (relight)	35.71	0.968	0.047	32.22	0.944	0.065
w/o indirect	34.68	0.968	0.047	32.28	0.947	0.058
Full	**35.87**	**0.971**	0.045	**32.46**	**0.948**	**0.057**

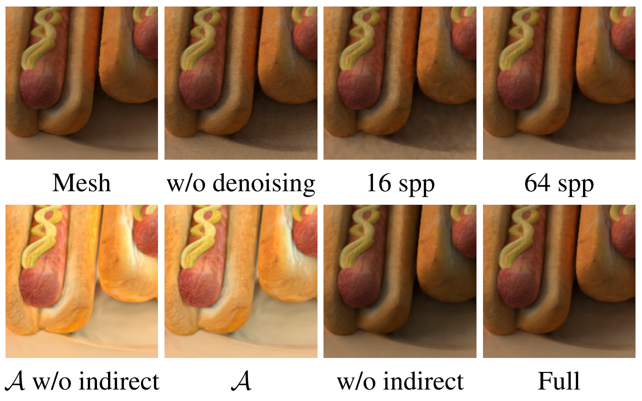

Figure 6. Ablation studies on different components of our method.

Tab. 4 and the qualitative comparisons are shown in Fig. 6, and we also give the predicted albedo maps $\mathcal{A}$ without and with indirect illumination during training in the third stage. We can see that the proposed GHPT can improve the relighting performance compared to the underlying meshes produced by the second stage, and the rendering results without spatial denoising and the results of 16 and 64 spp are noisier than those of 256 spp. Additionally, training without indirect illumination leads to overbrightening on the albedo in the concave region, and training and rendering both without indirect illumination results in overdarkening.

6. Conclusion

In this work, we introduce GHPT, a physically-based inverse rendering pipeline to create real-time relightable GS models. First, we generate multi-view consistent depth maps and reconstruct the surface mesh of the 3D scene using PGSR. Then physically-based differentiable rendering is performed on the mesh to obtain the textures that store material parameters and environment maps. Finally, FIPT is performed on the rasterized G-buffer, and visibility and indirect illumination are computed using the textured mesh in the previous stage, which leverages hardware-accelerated ray tracing to improve the rendering efficiency. Experimental results demonstrate that the relighting performance of our method outperforms all baselines, and our method can relight and composite Gaussian splatting in real-time. In the future, we will improve the accuracy of geometry reconstruction and material decomposition to achieve better inverse rendering and relighting performance.

References

[1] Sai Praveen Bangaru, Lifan Wu, Tzu-Mao Li, Jacob Munkberg, Gilbert Bernstein, Jonathan Ragan-Kelley, Frédo Durand, Aaron Lefohn, and Yong He. SLANG.D: Fast, modular and differentiable shader programming. *ACM TOG*, 42 (6), 2023. 5

[2] 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 5460–5469, 2022. 2, 5

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *ICCV*, pages 19640–19648, 2023. 2

[4] 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 *ECCV*, pages 294–311, 2020. 2

[5] Zoubin Bi, Yixin Zeng, Chong Zeng, Fan Pei, Xiang Feng, Kun Zhou, and Hongzhi Wu. GS^3: Efficient relighting with triple Gaussian splatting. In *ACM SIGGRAPH Asia*, page 12, 2024. 2, 1

[6] Blender Foundation. Blender 4.2.3 LTS, 2024. `https://www.blender.org`. 5

[7] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. PGSR: Planar-based Gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE TVCG*, 31(9):6100–6111, 2025. 2, 3

[8] Hongze Chen, Zehong Lin, and Jun Zhang. GI-GS: Global illumination decomposition on Gaussian splatting for inverse rendering. In *ICLR*, pages 75713–75742, 2025. 1, 2, 6, 7

[9] Yuxin Dai, Qi Wang, Jingsen Zhu, xi db, Yuchi Huo, Chen Qian, and Ying He. Inverse rendering using multi-bounce path tracing and reservoir sampling. In *ICLR*, pages 20526–20547, 2025. 2

[10] Kang Du, Zhihao Liang, and Zeyu Wang. GS-ID: Illumination decomposition on Gaussian splatting via adaptive light aggregation and diffusion-guided material priors. In *ICCV*, 2025. To appear. 1, 2

[11] Jiahui Fan, Fujun Luan, Jian Yang, Miloš Hašan, and Beibei Wang. RNG: Relightable neural Gaussians. In *CVPR*, pages 26525–26534, 2025. 2

[12] 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. In *ECCV*, pages 73–89, 2024. 1, 2, 3, 6, 7

[13] Chun Gu, Xiaofei Wei, Zixuan Zeng, Yuxuan Yao, and Li Zhang. IRGS: Inter-reflective Gaussian splatting with 2D Gaussian ray tracing. In *CVPR*, pages 10943–10952, 2025. 1, 2, 3, 6, 7

[14] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using Monte Carlo rendering and denoising. In *NeurIPS*, pages 22856–22869, 2022. 2, 5

[15] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH*, page 32, 2024. 2

[16] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.Jit: a just-in-time compiler for differentiable rendering. *ACM TOG*, 41(4):124, 2022. 2

[17] 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, 2023. 2, 5, 7, 8, 1

[18] James T. Kajiya. The rendering equation. *ACM SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986. 3

[19] Simon Kallweit, Chris Cummings, Benedikt Bitterli, Sai Bangaru, and Yong He. SlangPy, 2025. https://github.com/shader-slang/slangpy. 5

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):139, 2023. 1, 2

[21] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM TOG*, 37(6):222, 2018. 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, 2024. 1, 2, 6, 7

[23] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM TOG*, 38(6):228, 2019. 2

[24] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and SVBRDF recovery using differentiable Monte Carlo rendering. *Comput. Graph. Forum*, 40(4):101–113, 2021. 2

[25] 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, 2020. 2

[26] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3D Gaussian ray tracing: Fast tracing of particle scenes. *ACM TOG*, 43(6):232, 2024. 2

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):102, 2022. 2

[28] Jacob Munkberg, Wenzheng Chen, Jon Hasselgren, Alex Evans, Tianchang Shen, Thomas Müller, Jun Gao, and Sanja Fidler. Extracting triangular 3D models, materials, and lighting from images. In *CVPR*, pages 8270–8280, 2022. 2, 5

[29] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H. Kim. Practical SVBRDF acquisition of 3D objects with unstructured flash photography. *ACM TOG*, 37(6):267, 2018. 2

[30] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *ICCV*, pages 14315–14325, 2021. 2

[31] Thomas Smith. GPUSorting, 2024. `https://github.com/b0nes164/GPUSorting`. 7

[32] Thomas Smith. GPU Prefix Sums, 2024. https://github.com/b0nes164/GPUPrefixSums. 7

[33] Thomas Smith, Raph Levien, and John D. Owens. Decoupled fallback: A portable single-pass GPU scan. In *SPAA*, page 255–268, 2025. 6

[34] Cheng Sun, Guangyan Cai, Zhengqin Li, Kai Yan, Cheng Zhang, Carl Marshall, Jia-Bin Huang, Shuang Zhao, and Zhao Dong. Neural-PBIR reconstruction of shape, material, and illumination. In *ICCV*, pages 18000–18010, 2023. 2

[35] Hanxiao Sun, Yupeng Gao, Jin Xie, Jian Yang, and Beibei Wang. SVG-IR: Spatially-varying Gaussian splatting for inverse rendering. In *CVPR*, pages 16143–16152, 2025. 1, 2, 6, 7

[36] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *ACM SIGGRAPH*, page 419–428, 1995. 4

[37] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, pages 27171–27183, 2021. 3

[38] Liwen Wu, Rui Zhu, Mustafa B. Yaldiz, Yinhao Zhu, Hong Cai, Janarbek Matai, Fatih Porikli, Tzu-Mao Li, Manmohan Chandraker, and Ravi Ramamoorthi. Factorized inverse path tracing for efficient and accurate material-lighting estimation. In *ICCV*, pages 3825–3835, 2023. 2, 4

[39] Chongjie Ye, Lingteng Qiu, Xiaodong Gu, Qi Zuo, Yushuang Wu, Zilong Dong, Liefeng Bo, Yuliang Xiu, and Xiaoguang Han. StableNormal: Reducing diffusion variance for stable and sharp normal. *ACM TOG*, 43(6):250, 2024. 3

[40] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM TOG*, 43(6):271, 2024. 2

[41] Chong Zeng, Guojun Chen, Yue Dong, Pieter Peers, Hongzhi Wu, and Xin Tong. Relighting neural radiance fields with shadow and highlight hints. In *ACM SIGGRAPH*, page 73, 2023. 2

[42] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM TOG*, 39(4), 2020. 2

[43] 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 3578–3587, 2023. 2

[44] 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):237, 2021. 2, 7

[45] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *CVPR*, pages 18622–18631, 2022. 2, 5, 7, 8