DIB-R++: Learning to Disentangle Material from Lighting Using a Deferred Image-based Renderer

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the challenging problem of predicting intrinsic object properties from a single image by exploiting differentiable graphics renderers. Many previous learning-based approaches for inverse graphics adopt rasterization-based renderers and assume naive lighting and material models, which often fail to account for non-Lambertian, specular reflections commonly observed in the wild. In this work, we propose DIB-R++, a deferred, image-based renderer which supports these photorealistic effects by combining rasterization and ray-tracing, taking advantage of their respective strengths—speed and realism. Our renderer incorporates environmental lighting and spatially-varying material models to efficiently approximate light transport, either through direct estimation or via spherical basis functions. Compared to more advanced physics-based differentiable renderers leveraging path tracing, DIB-R++ is highly performant due to its compact and expressive shading model, which enables easy integration with learning frameworks for geometry, reflectance and lighting prediction from a single image without requiring any ground-truth. We experimentally demonstrate that our approach achieves superior material and lighting disentanglement on synthetic and real data compared to existing rasterization-based approaches and showcase several artistic applications including material editing and relighting.

1 Introduction

Inferring intrinsic 3D properties from 2D images is a long-standing goal of computer vision [1]. In recent years, differentiable rendering has shown great promise in estimating shape, reflectance and illumination from real photographs. Differentiable renderers have become natural candidates for learning-based inverse rendering applications, where image synthesis algorithms and neural networks can be jointly optimized to model physical aspects of objects from posed images, either by leveraging strong data priors or by directly modeling the interactions between light and surfaces.

Not all differentiable renderers are made equal. On the one hand, recent physics-based differentiable rendering techniques [2, 3, 4, 5, 6] try to model the full light transport with proper visibility gradients, but they tend to require careful initialization of scene parameters and typically exhibit high computational cost which limits their usage in larger end-to-end learning pipelines. On the other hand, performance-oriented differentiable renderers [7, 8, 9, 10] trade physical accuracy for scalability and speed by approximating scene elements through neural representations or by employing simpler shading models. While this latter line of work has found some success in 3D scene reconstruction, the frequent assumptions of Lambertian-only surfaces and low-frequency lighting prevent them from modeling more complex specular transport commonly observed in the real world.

In this work, we consider the problem of single-view 3D object reconstruction without any 3D supervision. To this end, we propose DIB-R++, a hybrid differentiable renderer that combines rasterization and ray-tracing through an efficient deferred rendering framework. Our framework

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

builds on top of DIB-R [7] and integrates physics-based lighting and material models to capture challenging non-Lambertian reflectance under unknown poses and illumination. Our method is versatile and supports both single-bounce ray-tracing and a spherical Gaussian representation for a compact approximation of direct illumination, allowing us to adapt and tune the shading model based on the radiometric complexity of the scene.

We validate our technique on both synthetic and real images and demonstrate superior performance on reconstructing realistic spatially-varying materials (SV-BRDFs) and lighting configurations over prior rasterization-based methods. We then follow the setting proposed in Zhang et al. [11] to show that DIB-R++ can reconstruct scene intrinsics also from real images without supervision, both quantitatively and visually. We further apply our framework to single-image appearance manipulation such as material editing and scene relighting.

2 Related Work

Differentiable Rendering. Research on differentiable rendering can be divided into two categories: physics-based methods focusing on photorealistic image quality, and approximate methods aiming at higher performance. The former differentiates the forward light transport simulation [2, 3, 4, 5, 6] with careful handling of geometric discontinuities. While capable of supporting global illumination, these techniques tend to be relatively slow to optimize or require a detailed initial description of the input in terms of geometry, materials, lighting and camera, which prevents their deployment in the wild. The latter line of works leverages simpler local shading models. Along this axis, rasterization-based differentiable renderers [7, 8, 9, 10] approximate gradients by generating derivatives from projected pixels to 3D parameters. These methods are restricted to primary visibility and ignore indirect lighting effects by construction, but their simplicity and efficiency offer an attractive trade-off for 3D reconstruction. We follow this line of work and build atop DIB-R [7] by augmenting its shading models with physics-based ones.

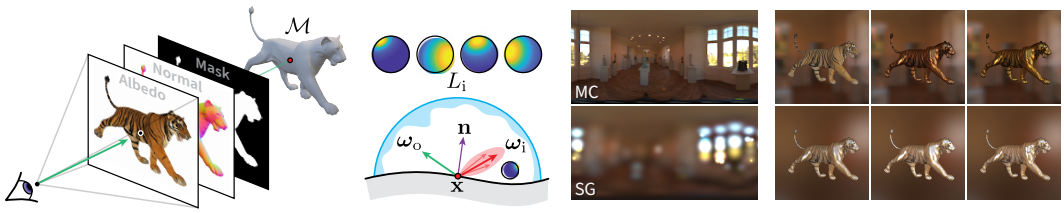
Learning-based Inverse Graphics. Recent research on inverse graphics targets the ill-posed problem of jointly estimating geometry, reflectance and illumination from image observations using neural networks. For single image inverse rendering, one dominant approach is to employ 2D CNNs to learn data-driven features and use synthetic data as supervision [12, 13, 14, 15], but these methods do not always generalize to complex real-world images [16]. To overcome this, a recent body of work investigates the use of self-supervised learning to recover scene intrinsics [17], including domain adaptation from synthetic reflectance dataset [18], object symmetry [19, 20], or multi-illumination images depicting the same scene [21, 22, 23, 24]. However, these methods either rely on specific priors or require data sources tedious to capture in practice. Most similar to us, DIB-R [7] tackles unsupervised inverse rendering in the context of differentiable rendering. Zhang et al. [11] further combines DIB-R with StyleGAN [25] generated images to extract and disentangle 3D knowledge. These works perform inverse rendering from real image collections without supervision, but may fail to capture complex material and lighting effects—in contrast, our method models these directly. Several techniques also try to handle more photorealistic effects but typically require more complex capture settings, such as known lighting and camera poses [26, 27, 28, 29], a co-located camera-flashlight setup [30, 31, 32], or densely captured multi-view images with lighting variation [33, 34]. In our work, we propose a hybrid differentiable renderer and learn to disentangle complex specular effects given a single image. Similar to the recent NeRD [33] and PhySG [34] which recover non-Lambertian reflectance and illumination with a spherical Gaussian (SG) basis [35], we also employ SGs to model the SV-BRDF and incident lighting, but apply this representation to mesh-based differentiable rendering with direct access to the surface.

3 Differentiable Deferred Image-based Rendering

In this section, we introduce DIB-R++, our differentiable rendering framework based on deferred shading [36]. DIB-R++ is a hybrid differentiable renderer that can efficiently approximate direct illumination and synthesize high-quality images. Concretely, our renderer leverages the differentiable rasterization framework of DIB-R [7] to recover shape attributes and further employs physics-based material and lighting models to estimate appearance.

3.1 Overview

We provide an overview of our technique in Fig. 1. Our method first rasterizes a 3D mesh to obtain diffuse albedo, surface normals, and a silhouette mask. This information is deferred to the shading pass, where outgoing radiance is either estimated stochastically or approximated using a spherical



(a) (b) (c) (d)

Figure 1: **Overview.** Given a 3D mesh $\mathcal{M}$, we employ (a) a rasterization-based renderer to obtain diffuse albedo, surface normals and mask maps. In the shading pass (b), we then use these buffers to compute the incident radiance by sampling or by representing lighting and the specular BRDF using a spherical Gaussian basis. Depending on the representation used (c), we can recover a wide gamut of specular/glossy appearances (d).

Gaussian basis. The rasterizer and shader are differentiable by design, allowing gradients to be simultaneously propagated to lighting, material and shape parameters for downstream learning tasks.

3.2 Background

Our goal is to provide a differentiable formulation of the rendering process to enable fast inverse rendering from 2D images. Let $\mathcal{M}$ be a 3D object in a virtual scene. We start from the (non-emissive) rendering equation (RE) [37], which states that the outgoing radiance L_o at any surface point $\mathbf{x} \in \mathcal{M}$ in the camera direction $\boldsymbol{\omega}_\mathrm{o}$ is given by

$$L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) = \int_{\mathcal{H}^2} f_\mathrm{r}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}) |\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}| \, \mathrm{d}\boldsymbol{\omega}_\mathrm{i}, \tag{1}$$

where L_i is the incident radiance, f_r is the (spatially-varying) bidirectional reflectance distribution function (SV-BRDF) and $\mathbf{n}$ is the surface normal at $\mathbf{x}$. The domain of integration is the unit hemisphere $\mathcal{H}^2$ of incoming light directions $\boldsymbol{\omega}_\mathrm{i}$. The BRDF characterizes the surface's response to illumination from different directions and is modulated by the cosine foreshortening term $|\mathbf{n} \cdot \boldsymbol{\omega}|$. Intuitively, Eq. (1) captures an energy balance and computes how much light is received and scattered at a shading point in a particular direction.

Estimating the RE typically requires Monte Carlo (MC) integration [38], which involves tracing rays from the camera into the scene. Albeit physically correct, this process is computationally expensive and does not generally admit a closed-form solution. MC estimators can exhibit high variance and may produce noisy pixel gradients at low sample count, which may significantly impact performance and convergence. To keep the problem tractable, we thus make several approximations of Eq. (1), which we detail in the next section.

3.3 Two-stage Deferred Rendering

We now describe our rendering framework (Fig. 1). We start by defining three families of parameters, where $\boldsymbol{\pi} \in \mathbb{R}^{d_\pi}$ encodes the shape attributes (e.g., vertex positions), $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ describes the material properties, and $\boldsymbol{\gamma} \in \mathbb{R}^{d_\gamma}$ captures the illumination in the scene. In what follows, we shall only consider a single pixel, indexed by p, within an RGB image $I \in \mathbb{R}_+^{3 \times h \times w}$ for notational simplicity.

Stage 1: Rasterization Pass. We first employ a differentiable rasterizer R [7] to generate primary rays $\boldsymbol{\omega}_\mathrm{o} \in \mathcal{S}^2$ from a camera and render our scene $\mathcal{M}$ into geometry buffers (commonly called *G-buffers*) containing the surface intersection point $\mathbf{x}_p \in \mathbb{R}^3$, the surface normal $\mathbf{n}_p \in \mathcal{S}^2$, and the spatially-varying material parameters $\boldsymbol{\theta}_p$ (e.g., diffuse albedo). This rendering pass also returns a visibility mask $v_p \in \{0, 1\}$ indicating whether pixel p is occupied by the rendered object, separating the foreground object I_f from its background environment I_b so that $I = I_\mathrm{f} + I_\mathrm{b}$. We have:

$$R(\mathcal{M}, p, \boldsymbol{\omega}_\mathrm{o}) = (\mathbf{x}_p, \mathbf{n}_p, \boldsymbol{\theta}_p, v_p). \tag{2}$$

Stage 2: Shading Pass. Given surface properties and outgoing direction $\boldsymbol{\omega}_\mathrm{o}$, we then approximate the outgoing radiance $L_\mathrm{o}(\mathbf{x}_p, \boldsymbol{\omega}_\mathrm{o})$ through several key assumptions. First, we restrict ourselves to direct illumination only (i.e. single-bounce scattering) and assume that the incoming radiance is given by a distant environment map $L_\mathrm{i} : \mathcal{S}^2 \to \mathbb{R}_+^3$. Therefore, we do not model self-occlusion and $L_\mathrm{i}(\mathbf{x}_p, \boldsymbol{\omega}_\mathrm{i}) \equiv L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i}; \boldsymbol{\gamma})$. Such simplification largely reduces computation and memory costs and is trivially differentiable. Second, we assume that the material parameters $\boldsymbol{\theta}$ can model both diffuse and specular view-dependent effects. At a high level, we define our shading model S so that:

$$S(\mathbf{x}_p, \mathbf{n}_p, \boldsymbol{\omega}_\mathrm{o}; \boldsymbol{\theta}_p, \boldsymbol{\gamma}) \approx L_\mathrm{o}(\mathbf{x}_p, \boldsymbol{\omega}_\mathrm{o}). \tag{3}$$

Importantly, a differentiable parameterization of S enables the computation of pixel gradients with respect to all scene parameters $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\gamma})$ by differentiating $I_p(\boldsymbol{\Theta}) = (S \circ R)(\mathcal{M}, p, \boldsymbol{\omega}_\mathrm{o})$. Given a scalar objective function defined on the rendered output I, $\partial I / \partial \boldsymbol{\pi}$ is computed using DIB-R [7]. In what follows, we thus mainly focus on formulating $\partial I / \partial \{\boldsymbol{\theta}, \boldsymbol{\gamma}\}$ so that all gradients can be computed using the chain rule, allowing for joint optimization of geometry, material and lighting parameters. We assume henceforth a fixed pixel p, and remove the subscript.

3.4 Shading Models

Since our primary goal is to capture a wide range of appearances, we provide two simple techniques to approximate Eq. (1): Monte Carlo (MC) and spherical Gaussians (SG). The former targets more mirror-like objects and can better approximate higher frequencies in the integrand, but is more expensive to compute. The latter is more robust to roughness variations but is limited by the number of basis elements. To model reflectance, we choose to use a simplified version of the isotropic Disney BRDF [39, 40] based on the Cook–Torrance model [41], which includes diffuse albedo $\mathbf{a} \in [0,1]^3$, specular albedo $s \in [0,1]$, surface roughness $\beta \in [0,1]$ and metalness $m \in [0,1]$. Metalness allows us to model both metals and plastics in a unified framework. We let the diffuse albedo vary spatially ($\mathbf{a} = \mathbf{a}(\mathbf{x})$) and *globally* define all other attributes to restrict the number of learnable parameters.

Monte Carlo Shading. Given a surface point $\mathbf{x} \in \mathcal{M}$ to shade, we importance sample the BRDF to obtain N light directions $\boldsymbol{\omega}_\mathrm{i}^k$ and compute the BRDF value. We represent the incident lighting $L_\mathrm{i}^{(\mathrm{MC})}$ as a high-dynamic range image $\boldsymbol{\gamma} \in \mathbb{R}_+^{3 \times h_l \times w_l}$ using an equirectangular projection, which can be queried for any direction via interpolation between nearby pixels. The final pixel color is then computed as the average over all samples, divided by the probability of sampling $\boldsymbol{\omega}_\mathrm{i}^k$:

$$S^{(\mathrm{MC})}(\mathbf{x}, \mathbf{n}, \boldsymbol{\omega}_\mathrm{o}; \boldsymbol{\theta}, \boldsymbol{\gamma}) = \frac{1}{N} \sum_{k=1}^{N} \frac{f_\mathrm{r}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}^k, \boldsymbol{\omega}_\mathrm{o}; \boldsymbol{\theta})\, L_\mathrm{i}^{(\mathrm{MC})}(\boldsymbol{\omega}_\mathrm{i}^k; \boldsymbol{\gamma})\, |\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}^k|}{p(\boldsymbol{\omega}_\mathrm{i}^k)}. \tag{4}$$

When the surface is near-specular (e.g., a mirror), one can efficiently estimate the RE as reflected rays are concentrated in bundles (e.g., to satisfy the law of reflection). However, this estimator can suffer from high variance for rougher surfaces; a higher number of samples may be necessary to produce usable gradients. This motivates the use of a more compact representation.

Spherical Gaussian Shading. To further accelerate rendering while preserving expressivity in our shading model, we use a spherical Gaussian (SG) [35] representation. Projecting both the cosine-weighted BRDF and incident radiance into an SG basis allows for fast, analytic integration within our differentiable shader, at the cost of some high frequency features in the integrand. Concretely, an SG kernel has the form $\mathcal{G}(\boldsymbol{\omega}; \boldsymbol{\xi}, \lambda, \boldsymbol{\mu}) = \boldsymbol{\mu}\, e^{\lambda(\boldsymbol{\xi} \cdot \boldsymbol{\omega} - 1)}$, where $\boldsymbol{\omega} \in \mathcal{S}^2$ is the input spherical direction to evaluate, $\boldsymbol{\xi} \in \mathcal{S}^2$ is the axis, $\lambda \in \mathbb{R}_+$ is the sharpness, and $\boldsymbol{\mu} \in \mathbb{R}_+^3$ is the amplitude of the lobe. We represent our environment map using a mixture of K lighting SGs $\mathcal{G}_l$, so that:

$$L_\mathrm{i}^{(\mathrm{SG})}(\boldsymbol{\omega}_\mathrm{i}; \boldsymbol{\gamma}) \approx \sum_{k=1}^{K} \mathcal{G}_l^k\big(\boldsymbol{\omega}_\mathrm{i}; \boldsymbol{\xi}_l^k, \lambda_l^k, \boldsymbol{\mu}_l^k\big), \tag{5}$$

where $\boldsymbol{\gamma} := \{\boldsymbol{\xi}_l^k, \lambda_l^k, \boldsymbol{\mu}_l^k\}_k$. For the BRDF, we follow Wang et al. [35] and fit a single, monochromatic SG to the specular lobe so that $f_\mathrm{r}^{(\mathrm{SG})}$ is a sum of diffuse and specular lobes. The full derivation can be found in our supplementary material. Finally, we approximate the cosine foreshortening term using a single SG $|\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}| \approx \mathcal{G}_c(\boldsymbol{\omega}_\mathrm{i}; \mathbf{n}, 2.133, 1.17)$ [42]. Regrouping all terms, the final pixel color can be computed as:

$$S^{(\mathrm{SG})}(\mathbf{x}, \mathbf{n}, \boldsymbol{\omega}_\mathrm{o}; \boldsymbol{\theta}, \boldsymbol{\gamma}) = \int_{\mathcal{S}^2} f_\mathrm{r}^{(\mathrm{SG})}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}^k, \boldsymbol{\omega}_\mathrm{o}; \boldsymbol{\theta})\, L_\mathrm{i}^{(\mathrm{SG})}(\boldsymbol{\omega}_\mathrm{i}; \boldsymbol{\gamma})\, \mathcal{G}_c(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\boldsymbol{\omega}_\mathrm{i}, \tag{6}$$

which has an analytic form that can be automatically differentiated inside our renderer. All parameters of the SGs, as well the BRDF parameters, are learnable.

Comparison. To visually compare our two shading techniques and understand their limitations, we render a unit sphere under the same lighting (represented differently) in Fig. 2. To do so, we first fit a HDR environment map using $K = 128$ SGs using an equirectangular projection. As shown on the left, SGs smooths out high frequency details and sharp corners but require much fewer parameters to reconstruct incident lighting (896 vs. 98 304). On the right, we visualize the effect of increasing the surface roughness β under the corresponding light representation.

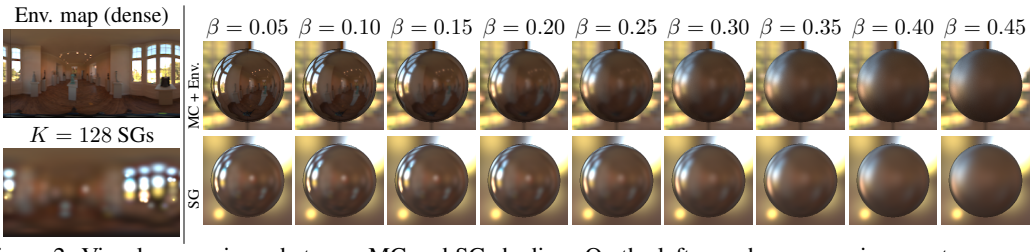


Figure 2: Visual comparisons between MC and SG shading. On the left, we show an environment map and its SG-representation ($K = 128$). While losing sharp details, SGs only needs 1% parameters compared to the dense pixel HDR map. On the right, we increase roughness left-to-right, revealing that our spherical basis can correctly approximate direct illumination at moderate-to-high roughness.

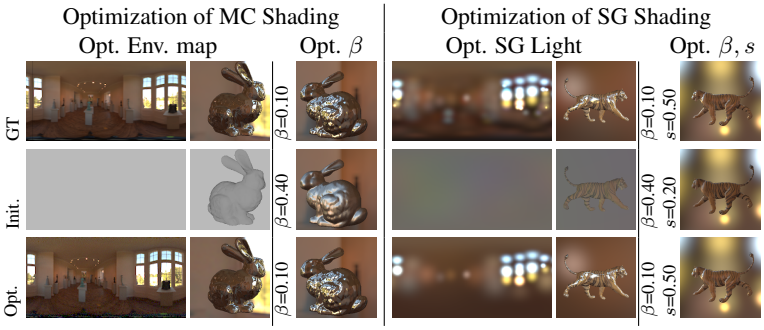


Figure 3: **Optimization**. We optimize complex lighting and material for both MC shading (left) and SG shading (right). In each case, we first optimize lighting and show the light and rendered image in the first block. Next we optimize the surface material and show the surface material value and rendered image in the second block.

Intuitively, this point of diminishing return indicates that MC is only so useful when the surface reflects most of the incoming light (e.g., a mirror). Indeed, when β is small enough ($\beta \to 0$) and we deal with a highly non-Lambertian surface, a small number of MC samples are enough to estimate direct illumination, which in turn implies faster render times and low memory cost. On the other end of the spectrum ($\beta \to 1$), significantly more samples (e.g., $N > 1000$) are needed to accurately integrate incident light, resulting in longer inference times. In such a case, SGs should be favored since they offer a significant improvement. In the absence of any prior knowledge on the material type, SG shading is preferred. This is reflected in our experiments in Sec. 5-6.

Optimization. We perform a sanity check on our renderer in Fig. 3 by optimizing for lighting and reflectance properties from a multi-view image L_1-loss with fixed geometry. We show the ground-truth (GT) parameters and rendered images in the first row, along with initial parameters (Init.) in the second row. Here, we optimize parameter separately using gradient-descent while the others are kept fixed to validate each component of our shading model. DIB-R++ can successfully estimate material and lighting parameters, including the environmental lighting, surface roughness and specular albedo of the object. We find that the converged material parameters closely match GT, while the optimized environment map loses some details due to gradients coming entirely from surface reflections (foreground supervision only). In particular, surface highlights are well captured.

4 Application: Single Image 3D Reconstruction

We demonstrate the effectiveness of our deferred framework through the learning-based problem of single image 3D reconstruction without supervision. While previous works [7, 11] generally focus on diffuse illumination only, our goal is to jointly infer geometry, reflectance, and lighting from a single image $\tilde{I}$ containing strong specular transport. To this end, we employ a convolutional neural network F, parameterized by learnable weights $\boldsymbol{\vartheta}$, to predict 3D attributes of a mesh $\mathcal{M}$ with pre-determined topology (sphere in our case). We adopt the U-Net [43] architecture of the original DIB-R [7, 11] and modify its output to also predict the appropriate BRDF attributes $\boldsymbol{\theta}$ and light parameters $\boldsymbol{\gamma}$ (pixel colors or SG coefficients) so that $F(\tilde{I}; \boldsymbol{\vartheta}) = (\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\gamma})$. We then render these parameters back to an image I using our deferred differentiable renderer and apply a loss $\mathcal{L}$ on the RGB output to compare the input image $\tilde{I}$ and the rendered image I, where:

$$\mathcal{L}(\boldsymbol{\vartheta}) = \alpha_{\text{im}}\mathcal{L}_{\text{im}}(\tilde{I}, I) + \alpha_{\text{msk}}\mathcal{L}_{\text{msk}}(\tilde{V}, V) + \alpha_{\text{per}}\mathcal{L}_{\text{per}}(\tilde{I}, I) + \alpha_{\text{lap}}\mathcal{L}_{\text{lap}}(\boldsymbol{\pi}) + \alpha_{\text{sm}}\mathcal{L}_{\text{sm}}(\boldsymbol{\pi}) + \alpha_{\text{mov}}\mathcal{L}_{\text{mov}}(\boldsymbol{\pi}). \tag{7}$$

Similar to DIB-R [7], we combine multiple consistency losses with regularizations: $\mathcal{L}_{\text{im}}$ is an image loss computing the L^1-distance between the rendered image and the input image, $\mathcal{L}_{\text{msk}}$ is an Intersection-over-Union (IoU) loss of the rendered silhouette V and the input mask $\tilde{V}$ of the object [10], $\mathcal{L}_{\text{per}}$ is a perceptual loss [44, 45] computing the L^1-distance between the pre-trained AlexNet [46] feature maps of rendered image and input image, $\mathcal{L}_{\text{lap}}$ is a Laplacian loss to penalize

the change in relative positions of neighboring vertices, $\mathcal{L}_{\text{sm}}$ is a smoothness loss to penalize the difference in normals for neighboring faces, and $\mathcal{L}_{\text{mov}}$ is a deformation loss to penalize the magnitude of predicted vertex deformation from the template. We set $\alpha_{\text{im}} = 20$, $\alpha_{\text{msk}} = 5$, $\alpha_{\text{per}} = 0.5$, $\alpha_{\text{lap}} = 5$, $\alpha_{\text{sm}} = 5$, and $\alpha_{\text{mov}} = 2.5$, which we empirically found worked best.

5 Evaluation on Synthetic Datasets

We conduct extensive experiments to evaluate the performance of DIB-R++. We first quantitatively evaluate on synthetic data where we have access to ground-truth geometry, material and lighting. Since MC ad SG shading have individual pros and cons, we validate them under different settings. In particular, we generate datasets with two different surface materials: purely **metallic** surfaces with no roughness, and **glossy** surfaces with random positive roughness. We compare the performance of both shading models against the baseline method [7].

Synthetic Datasets. We chose 485 different car models from *TurboSquid*[1] to prepare data for metallic and glossy surfaces. We also collected 438 freely available high-dynamic range (HDR) environment maps from *HDRI Haven*[2] to use as reference lighting, which contain a wide variety of illumination configurations for both indoor and outdoor scenes. To render all 3D models, we use Blender's Cycles[3] path tracer with the Principled BRDF model [39]. We create two datasets, Metallic-Surfaces and Glossy-Surfaces. For metallic surfaces, we set $\beta = 0$ and $m = 1$. Conversely, we set $m = 0$, $s = 1$ and randomly pick $\beta \in [0, 0.5]$ to generate images for glossy surfaces. Please refer to our supplementary for a detailed overview of our synthetic dataset.

Baseline. We compare our method with the rasterization-based baseline DIB-R [7], which supports spherical harmonics (SH) lighting. While the original lighting implementation in [7] is monochromatic, we extend it to RGB for a fairer comparison. For quantitative evaluation, we first report the common L^1 pixel loss between the re-rendered image using our predictions and ground-truth (GT) image, and 2D IoU loss between rendered silhouettes and ground-truth masks. We experimentally find that these numbers are very close in different methods. Thus we further evaluate the quality of diffuse albedo, specular material and lighting predictions using normalized cross correlation (NCC). We provide more details of these metrics in the supplementary material.

5.1 Metallic Surfaces

Experimental Settings. We first apply all methods to the metallic car dataset. Since this surface property is known as a priori, we relax the task for MC shading by setting $\beta = 0$ and only predict geometry, diffuse albedo and lighting from the input image. This allows us to render MC at a low sample count ($N = 4$), achieving higher rendering speed and a lower memory cost. In particular, we predict the relative offset for all $|\mathcal{M}| = 642$ vertices in a mesh and a 256×256 texture map, following the choices in [7]. We also predict a 256×256 RGB environment map. For SG shading, we predict all parameters. While shape and texture are the same as MC shading, we adopt $K = 32$ for SG and predict two global parameters β and s for the specular BRDF. This keeps the number of parameters relatively low while providing enough flexibility to capture different radiometric configurations.

Experimental Results. Quantitative and qualitative results are shown in Fig. 4 and Table 1 (Left), respectively. Since the main loss function comes from the difference between GT and re-rendered images, we find the re-rendered images (with light) from the predictions are all close to the GT image for different methods, and quantitatively, the image loss and 2D IoU loss are also similar across different models. However, we observe significant differences on the predicted albedo and lighting. Specifically, in Fig. 4, MC shading successfully predicts cleaner diffuse albedo maps and more accurate light, while Chen et al. [7] "bakes in" high specular effects into the texture. Quantitatively, we outperform [7] with a $3\times$ improvement in terms of NCC for lighting, demonstrating the effectiveness of our DIB-R++. We further compare MC shading with SG shading.

Shading	Metallic surfaces (↓)				Glossy surfaces (↓)			
	Image	2D IoU	Light	Tex.	Image	2D IoU	Light	Tex.
MC	0.021	0.040	**0.053**	**0.145**	0.021	0.038	0.080	0.141
SG	0.021	0.037	0.065	0.203	0.020	0.041	**0.076**	**0.123**
SH [7]	0.020	0.040	0.183	0.201	0.020	0.041	0.115	0.136

Table 1: Quantitative results of single image 3D Reconstruction on synthetic data. While all the methods achieve comparable performance on re-rendered images and 2D IoUs, both MC and SG achieve better results on lighting and texture. MC is particularly better for metallic surfaces, and SG works best for glossy surfaces.

[1] `https://turbosquid.com`. We obtain consent via agreement with TurboSquid, following their license at `https://blog.turbosquid.com/turbosquid-3d-model-license`.

[2] `https://hdrihaven.com`. We follow the CCO license at `https://hdrihaven.com/p/license.php`.

[3] `https://blender.org`

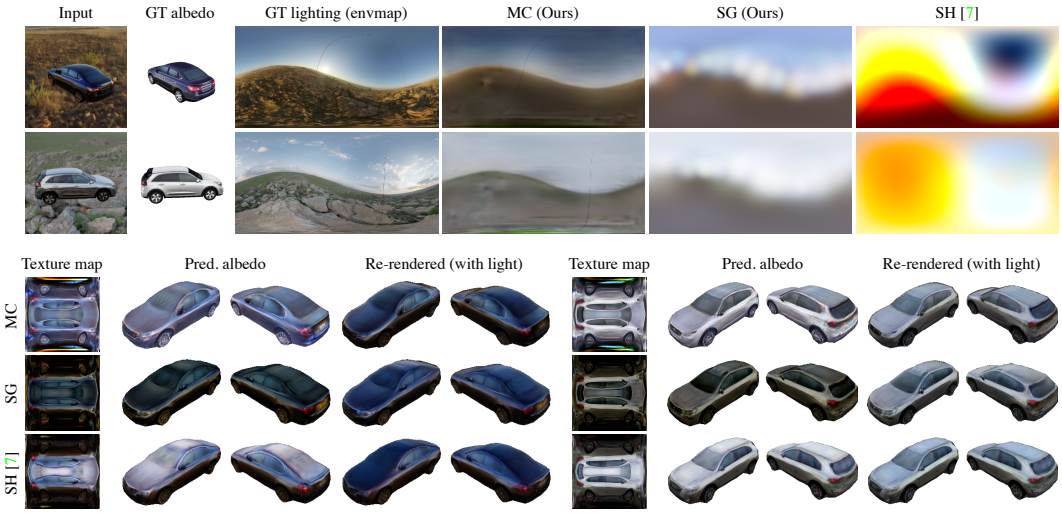


Figure 4: Prediction results on the **metallic** car dataset ($\beta = 0$). While all re-rendered images look almost identical, the underlying radiometric components (material and lighting) are different. A SH basis [7] cannot recover the high frequency details of the sky light maps. In this case, MC performs best due to low variance in the estimator for mirror-like BRDFs. SGs can recover the overall form and contrast of the light map, but tend to predict incorrect texture maps incorporating the ground dominant color (e.g., brown).

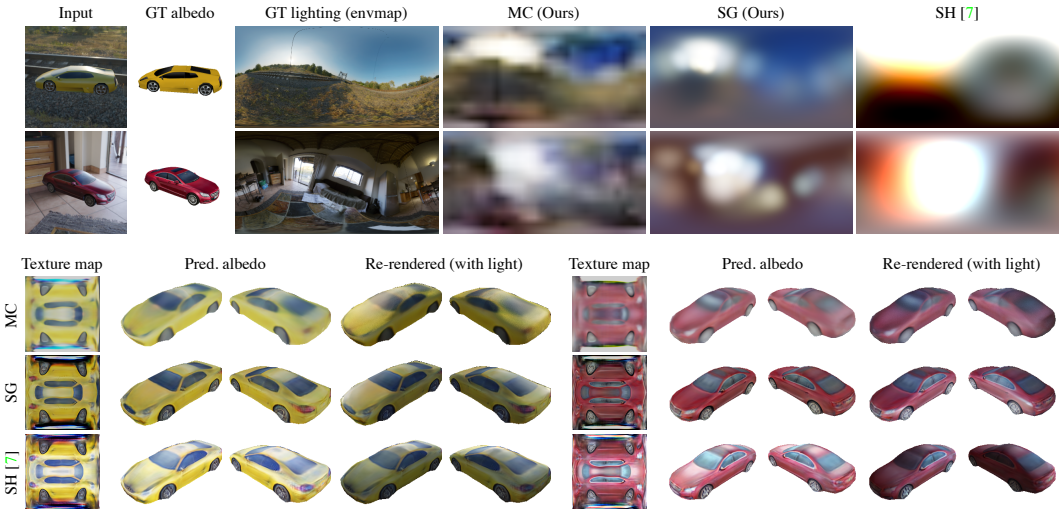


Figure 5: Prediction results on the **glossy** car dataset ($\beta > 0$). When the objects are glossy but not perfectly specular, SGs can correctly disentangle reflectance from lighting, as evidenced by the absence of white highlights in the predict albedos. Chen et al. [7] cannot capture these bright regions due to a diffuse-only shading model, while MC oversmooths the predictions due to noisier pixel gradients. While all methods cannot completely reconstruct the environment map, our method can predict the correct dominant light location and sky color.

While SG shading achieves reasonable lighting predictions, it fails to reconstruct the high frequency details in the lighting and has circular spot effects caused by the isotropic SGs.

5.2 Glossy Surfaces

Experimental Settings. We further apply our model to synthetic images rendered with positive roughness (glossy). To apply MC shading in such a case, we assume no prior knowledge for material and use a high sample count ($N = 1024$) to account for possibly low roughness images in the dataset. Due to high rendering time and memory cost, we subsample 4% of the pixels and apply $\mathcal{L}_{\text{im}}$ to those pixels only in each training iteration. As such, we do not use the perceptual loss $\mathcal{L}_{\text{per}}$ as it relies on the whole image. We predict a 32×16 environment map for lighting and predict global β and m for the specular BRDF. For SG shading, we use the same settings as for the metallic surfaces.

Experimental Results. We apply both MC and SG shading and compare with [7]. Results are shown in Fig. 5 and Table 1 (Right). Qualitatively, SG shading has better lighting predictions with correct high-luminance regions. The specular highlights in the images successfully guide SG shading, while

the bright reflection on the car roof and front cover are fused with texture map in [7]. MC also has reasonable lighting predictions, but the predicted environment map lacks structure due to weak surface reflections. Without a perceptual loss term, the predicted textures also tend to be blurrier.

Quantitatively, SG shading significantly outperforms [7] on lighting prediction in terms of NCC (0.078 vs. 0.127) and improves on texture prediction in terms of BRDF/lighting disentanglement. We also compare with MC shading, where MC achieves slightly worse results on lighting predictions compared to SG, but is still much better than [7].

5.3 Discussion

As shown in our previous two experiments, Monte Carlo shading works best under a metallic assumption ($\beta = 0$), in which case the rendered images can have rich details at low sample count ($N \leq 4$). However, when the surface is more Lambertian (i.e., when β is becoming larger), we have to compensate with a larger N to produce noise-free renderings, which impacts learning both time- and memory-wise. As a consequence, we recommend applying MC shading to metallic surfaces only, and default to SGs otherwise.

Our spherical Gaussian shading pipeline provides an analytic formulation for estimating the rendering equation, which avoids the need of tracing ray samples, largely accelerating the rendering process. While SGs can be blurry on metallic surfaces, in most case (e.g., when $\beta \geq 0.2$) it can model similar rendering effects at a fraction of the cost, achieving better results than MC shading and [7].

After inspecting the predicted surface material properties (β, s, m) with the ground-truth parameters in Blender, we find little correlation. As for SG, we are using only 32 basis elements to simulate a complex, high definition environment map (2K). Since SGs can only represent a finite amount of details, we find the predicted global β tends to be too small. One hypothesis for this is that the optimizer artificially prefers more reflections (and thus lower roughness) to be able to estimate at least some portions of the environment map. On the other hand, in MC, due to the absence of a perceptual loss, the predicted texture is very smooth and cannot represent GT to a high detail. We find that the predicted β and m do not have strong correlation with the GT material.

In summary, when we have no prior knowledge about the material, our re-rendered images can be very close to the input images but the predicted material parameters are not always aligned with the GT materials. We believe this problem can be relieved by incorporating additional local constraints, e.g., part-based material priors, or by leveraging anisotropic SGs [47]. For instance, a car body is metallic while its wheels are typically diffuse; predicting different parameters for each region has the potential of improving disentanglement and interpretability. We leave this as future work.

6 Evaluation on a Real-world Dataset

We further qualitatively evaluate our method via training on StyleGAN generated data, following the pipeline in [11] in Sec. 6.1, and use our predictions to perform artistic manipulation in Sec. 6.2.

6.1 Realistic Imagery

Experimental Settings. Our DIB-R++ can also be applied to learn 3D properties from realistic imagery. Following [11], we use StyleGAN [25] to generate multi-view images of cars and faces, which is the data we need to train our model. The generated objects contains cars under various lighting conditions, ranging from high specular paint to nearly diffuse. Thus, we only apply SG shading and adopt the same setting, where we predict $|\mathcal{M}| = 642$ vertex movements, a 256×256 diffuse texture map, $K = 32$ SG bases and two global β and s. We also compare [11] as the baseline on the same dataset by using the same training procedure.

Results. In the absence of groundtruth on the StyleGAN generated data, we qualitatively evaluate our results and compare with [11] in Fig. 6. Our DIB-R++ reconstructs more faithful material and lighting components, producing an interpretable decomposition. Specifically, our model can represent the dominant light direction more accurately, while naive shading tend to merge reflectance with lighting. We also provide an example with monochromatic lighting on a face example to reduce the degrees of freedom for the SH representation, yet it cannot correctly model light. More comparisons can be found in the supplementary.

6.2 Material Editing and Relighting

Finally, we demonstrate some applications of DIB-R++ to artistic manipulation in Fig. 7. On the left, we show examples of editing the diffuse albedo, where we can insert text, decals or modify

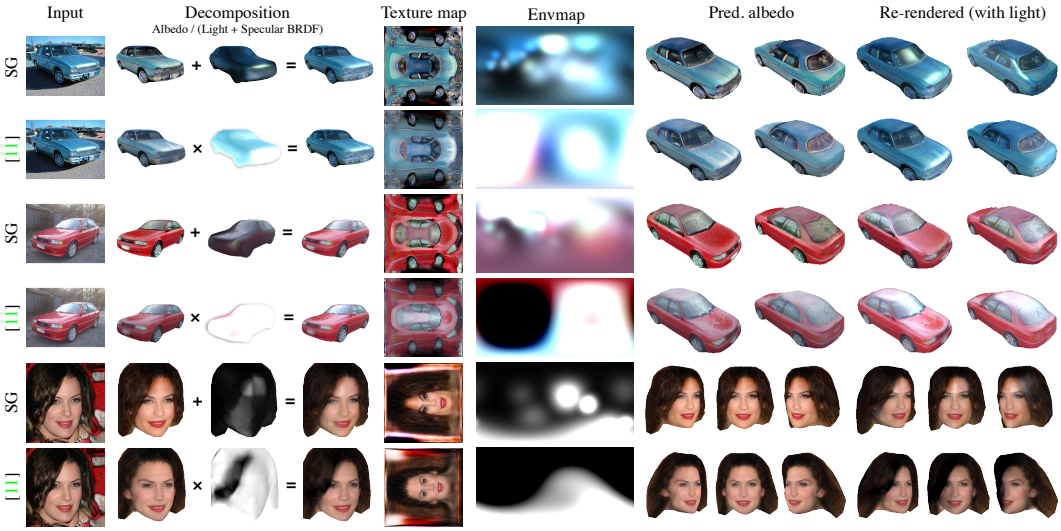


Figure 6: Results on real imagery from the StyleGAN-generated dataset (cars and white female faces). Our method can recover a meaningful decomposition as opposed to [11], as shown by cleaner texture maps and directional highlights (e.g., car windshield). Even when using monochromatic lighting on faces, our method can correctly predict the specular highlights on the forehead and none in the hair, while SH produces dark artifacts.

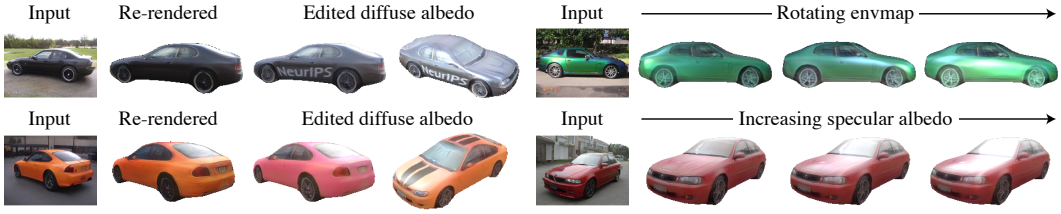


Figure 7: Our SG method allows for artistic manipulation of appearance, such as novel view synthesis, material editing (both diffuse and specular components), and relighting, thanks to our effective disentanglement.

the base tint. Since our textures are not contaminated by lighting, clean texture maps can be easily edited by hand and the re-rendered images look natural. On the right, we show examples of editing lighting and surface materials, where we rotate the light (top) or increase glossiness (bottom). We also showcase results where we change lighting orientation or modify the object's glossiness with consistent shading, which is not feasible with a naive, Lambertian-only shading model.

7 Conclusion

We presented DIB-R++, a hybrid differentiable renderer that can effectively disentangle material and lighting. When embedded in a learning framework for single image 3D reconstruction, our method produces state-of-the-art results, and enables applications such as material editing and relighting. Our method is not perfect and the predicted base color may sometimes "bleed" into the lighting predictions. Combining our technique with segmentation methods like DatasetGAN [48] could alleviate this issue for a more practical, artist-friendly disentanglement.

Broader Impact. Our work focuses on disentangling geometry from appearance using a differentiable renderer, a relatively nascent research area. We show that augmenting a rasterization-based renderer with physics-based shading models improves reconstruction and allows for easier integration within larger machine learning pipelines. DIB-R++ relies on simple topology and strong data prior assumptions to produce useful decompositions; therefore it cannot generalize to the complexity and multi-modality of real-world scenes in its current form. Nonetheless, we believe that our work takes an important step in the joint estimation of shape, material and environmental lighting from a single image and we hope that it can advance applications in performance-oriented settings such as AR/VR, simulation technology and robotics. For instance, autonomous vehicles need to correctly assess their surroundings from limited signals; directly modeling light-surface interactions (e.g., specular highlights) may provide important cues to this end. Like any ML model, DIB-R++ is prone to biases imparted through training data which requires an abundance of caution when applied to sensitive applications (e.g., human faces). For purposes of real deployment, one would need to carefully inspect and de-bias the dataset to depict the target distribution of a wide range of possible skin tones.

References

[1] Harry Barrow, J. Tenenbaum, A. Hanson, and E. Riseman. Recovering intrinsic scene characteristics. *Comput. Vis. Syst*, 2:3–26, 1978.

[2] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[3] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 38(6), November 2019.

[4] Sai Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):245:1–245:18, 2020.

[5] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 39(4), July 2020.

[6] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 39(4):143:1–143:19, 2020.

[7] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3D objects with an interpolation-based differentiable renderer. In *Advances In Neural Information Processing Systems (NeurIPS)*, 2019.

[8] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6), 2020.

[9] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[10] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] Yuxuan Zhang, Wenzheng Chen, Jun Gao, Huan Ling, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image GANs meet differentiable rendering for inverse graphics and interpretable 3D neural rendering. In *International Conference on Learning Representations (ICLR)*, 2020.

[12] Takuya Narihira, Michael Maire, and Stella X Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 2992–2992, 2015.

[13] Jian Shi, Yue Dong, Hao Su, and Stella X. Yu. Learning non-Lambertian object intrinsics across ShapeNet categories. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1685–1694, 2017.

[14] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):1–11, 2018.

[15] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik Lensch, and Jan Kautz. Two-shot spatially-varying BRDF and shape estimation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3982–3991, 2020.

[16] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Deep hybrid real and synthetic training for intrinsic decomposition. In *Eurographics Symposium on Rendering: Experimental Ideas & Implementations (EGSR)*, 2018.

[17] Hassan Alhaija, Siva Mustikovela, Varun Jampani, Justus Thies, Matthias Niessner, Andreas Geiger, and Carsten Rother. Intrinsic autoencoders for joint neural rendering and intrinsic image decomposition. In *International Conference on 3D Vision (3DV)*, 2020.

[18] Yunfei Liu, Yu Li, Shaodi You, and Feng Lu. Unsupervised learning for intrinsic image decomposition from a single image. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[19] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3D objects from images in the wild. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10, 2020.

[20] Shangzhe Wu, Ameesh Makadia, Jiajun Wu, Noah Snavely, Richard Tucker, and Angjoo Kanazawa. De-rendering the world's revolutionary artefacts. *arXiv preprint arXiv:2104.03954*, 2021.

[21] Louis Lettry, Kenneth Vanhoey, and Luc Van Gool. Unsupervised deep single-image intrinsic decomposition using illumination-varying image sequences. In *Computer Graphics Forum*, volume 37, pages 409–419, 2018.

[22] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9039–9048, 2018.

[23] Wei-Chiu Ma, Hang Chu, Bolei Zhou, Raquel Urtasun, and Antonio Torralba. Single image intrinsic decomposition without a single intrinsic image. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 201–217, 2018.

[24] Ye Yu and William AP Smith. Inverserendernet: Learning single image inverse rendering. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019.

[26] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Proc. of the 12th Eurographics Conference on Rendering*, page 103–114, 2001.

[27] Hendrik P.A. Lensch, Jochen Lang, Asla M. Sa, and Hans-Peter Seidel. Planned sampling of spatially varying BRDFs. *Computer Graphics Forum*, 2003.

[28] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 33(6), November 2014.

[29] Rui Xia, Yue Dong, Pieter Peers, and Xin Tong. Recovering shape and spatially-varying surface reflectance under unknown illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 35(6), 2016.

[30] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. Practical SVBRDF acquisition of 3D objects with unstructured flash photography. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):1–12, 2018.

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

[32] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, and Ravi Ramamoorthi David Kriegman. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images, 2020.

[33] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik PA Lensch. NeRD: Neural reflectance decomposition from image collections, 2020.

[34] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. PhySG: Inverse rendering with spherical gaussians forphysics-based material editing and relighting. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[35] Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 28(5):1–10, December 2009.

[36] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: A vlsi system for high performance graphics. In *Proc. of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, page 21–30, 1988.

[37] James T. Kajiya. The rendering equation. SIGGRAPH '86, page 143–150, 1986.

[38] Matt Pharr, Greg Humphreys, and Wenzel Jakob. *Physically Based Rendering, Third Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., 3rd edition, 2016.

[39] Brent Burley. Physically-based shading at Disney. In *Practical Physically-Based Shading in Film and Game Production (SIGGRAPH Course)*, pages 10:1–10:7, 2012.

[40] Brian Karis (Epic Games). Real shading in Unreal Engine 4. In *ACM SIGGRAPH 2013 Courses*, 2013.

[41] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, January 1982.

[42] Julian Meder and Beat Bruderlin. Hemispherical Gaussians for accurate light integration. *Computer Vision and Graphics*, 2018.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241, 2015.

[44] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[47] Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. Anisotropic spherical gaussians. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 32(6), November 2013.

[48] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. DatasetGAN: Efficient labeled data factory with minimal human effort. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We provided extensive experiment in Sec. 5, 6.1 and 6.2.
   (b) Did you describe the limitations of your work? [Yes] See Sec. 7 and Broader Impact.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We provided the discussion in the Broader Impact section with further discussion in Supplementary Material.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The current implementation requires complex dependencies. We plan to release the code after refactoring.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See corresponding sections in main paper (Sec. 5, 6.1 and 6.2) and Supplementary Material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] According to our experiments, our model converged well, and random seeds had little effect on performance.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] This is provided in the Supplementary Material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We provided the references to the datasets in Sec. 5 and 6.1.
   (b) Did you mention the license of the assets? [Yes] We provided the license of *TurboSquid* and *HDRI Haven* datasets in Sec. 5.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We discussed this in Sec. 6.1.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We provided discussion on this in the Supplementary Material.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]