SplitNeRF: Split Sum Approximation Neural Field for Joint Geometry, Illumination, and Material Estimation

Jesus Zarzar
KAUST

Bernard Ghanem
KAUST

Abstract

We present a novel approach for digitizing real-world objects by estimating their geometry, material properties, and environmental lighting from a set of posed images with fixed lighting. Our method incorporates into Neural Radiance Field (NeRF) pipelines the split sum approximation used with image-based lighting for real-time physical-based rendering. We propose modeling the scene's lighting with a single scene-specific MLP representing pre-integrated image-based lighting at arbitrary resolutions. We achieve accurate modeling of pre-integrated lighting by exploiting a novel regularizer based on efficient Monte Carlo sampling. Additionally, we propose a new method of supervising self-occlusion predictions by exploiting a similar regularizer based on Monte Carlo sampling. Experimental results demonstrate the efficiency and effectiveness of our approach in estimating scene geometry, material properties, and lighting. Our method is capable of attaining state-of-the-art relighting quality after only $\sim$1 *hour of training in a single NVIDIA A100 GPU.*

1. Introduction

The idea of creating realistic and immersive digital environments has piqued the imagination of countless science fiction authors, science fiction directors, and scientists. In the past few years, the fields of computer graphics and computer vision have advanced so much that we are capable of creating photo-realistic environments [6, 21, 26], as well as capturing real-world environments in a way that allows us to render new photo-realistic views [24, 29]. However, the creation of digital twins [9] of objects that can be integrated within photo-realistic environments still requires artists to meticulously hand-design realistic object meshes, materials, and lighting. While this is feasible for generating a few scenes, large-scale digitization requires automatic ways of reconstructing real-world objects along with their corresponding material properties.

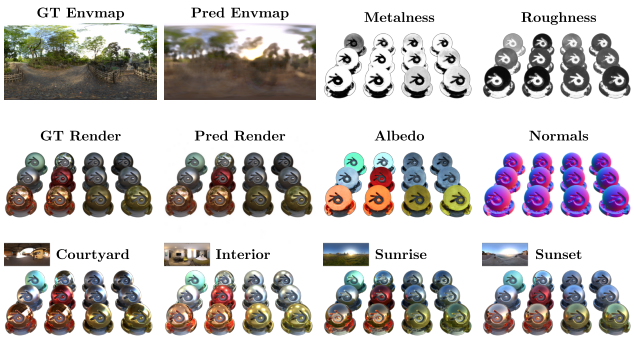

Figure 1. We visualize the lighting, material properties, and geometry predicted by our model in addition to several relighting predictions of the 'materials' scene. Our method is capable of simultaneously predicting high-frequency illumination, material properties (albedo, metalness, and roughness), and geometry.

In this work, we address the problem of object inverse rendering: extracting object geometry, material properties, and environment lighting from a set of posed images of the object. Inverse rendering enables the seamless integration of virtual objects into different environments with varying illumination conditions from simple image captures taken by commonplace camera sensors.

Neural rendering methods, such as Neural Radiance Fields (NeRF) [1, 18, 31], have revolutionized novel view synthesis, 3D reconstruction from images, and inverse rendering. By directly modeling outgoing radiance at each point in 3D space, NeRF methods excel at accurately recovering scene geometry and synthesizing novel views. However, a drawback of this approach is that the learned radiance representation entangles environment lighting with the rendered scene's properties, making it challenging to recover material properties and illumination. Due to the success of NeRFs in reconstructing scenes, several works have proposed modifications to enable inverse rendering [3, 16, 27]. These works build upon NeRF by decomposing radiance into a function of illumination and material properties but differ in their ways of modeling lighting and reflections.

We follow suit with the main goal of efficiency without sacrificing reconstruction quality or the ability to recover high-frequency illumination details.

To achieve these goals, we rely on the split sum approximation [11], which is commonly used in efficient image-based lighting techniques and has been successfully applied for inverse rendering before [3, 20]. This approximation involves splitting the surface reflectance equation into two factors: one responsible for pre-integrating illumination and the other for integrating material properties. Our first key insight is that this separation allows us to estimate pre-integrated illumination using a Multi-Layer Perceptron (MLP). This manner of modeling the pre-integrated illumination function is inspired by the modeling of radiance fields, which model a complex integral of lighting and material properties using an MLP. Correspondingly, our illumination representation inherits beneficial properties observed with the modeling of radiance fields such as smoothness. To ensure accurate learning of illumination, we introduce a novel regularizer based on Monte Carlo sampling.

However, the split sum approximation on its own does not take into account self-occlusions. This hinders the estimation of material properties since shadows tend to be incorrectly attributed to being part of an object's albedo. Thus, we derive an occlusion factor to correctly account for self-occlusions. This factor is then approximated via Monte Carlo sampling and used to supervise an MLP dedicated to predicting self-occlusions.

Altogether, our method is capable of attaining state-of-the-art relighting results with under an hour of training on a single NVIDIA A100 GPU.

Contributions. We claim the following contributions:

(i) We propose a novel representation for representing pre-integrated illumination as a single MLP along with a corresponding regularization to ensure accurate learning.

(ii) We derive a method for approximating the effect of self-occlusions on pre-integrated lighting and use it to supervise an occlusion MLP.

(iii) We demonstrate the effectiveness of our method in extracting environmental lighting and material properties, achieving state-of-the-art relighting quality with under one hour of training on a single NVIDIA A100 GPU.

2. Related Work

The problem of digitizing real-world objects and environments has long been a subject of active research in computer vision and computer graphics. We approach this problem through the lenses of neural rendering and neural inverse rendering; paradigms with lots of recent attention. We now provide a brief overview of related works in these areas.

2.1. Neural Rendering and 3D Reconstruction

Novel view synthesis is the task of rendering new views of a scene given a set of observations of the scene. Neural Radiance Fields (NeRF) [18] and its variants [1, 4, 19, 25, 31] have demonstrated remarkable success in the task of novel view synthesis. NeRF directly models the volumetric scene function by predicting radiance and density at each 3D point in space while supervising learning with a photometric reconstruction loss. Due to its success in implicitly learning accurate 3D reconstructions, several works have branched out to reconstruct accurate meshes through neural rendering [22, 28]. Signed Distance Function (SDF)-based methods [12, 33, 34, 37] model density as a function of the SDF to obtain well-defined surfaces. By increasing sharpness during training in the conversion from SDF to density these methods can transition from volume rendering to surface rendering as they train. While effective, these methods suffer from entangled representations of scene geometry, material properties, and lighting. Our work follows the surface rendering pipeline proposed in [33], but reformulates the radiance prediction in a manner that disentangles environment lighting and material properties.

2.2. Neural Inverse Rendering

The task of inverse rendering consists of estimating the properties of a 3D scene such as shape, material, and lighting from a set of image observations, and is a long-standing problem in computer graphics. The success of neural rendering methods for novel view rendering and 3D reconstruction has led to a variety of works [2, 3, 14, 16, 20, 27, 38, 39, 41, 42] exploiting neural rendering for inverse rendering. Due to the challenging nature of this problem, a wide variety of simplifying assumptions have been adopted. Some works simplify the modeling of lighting by using low-frequency representations such as spherical gaussians [2, 10, 27, 36, 38, 41, 42] or low-resolution environment maps [2, 38, 42]. While this approximation generally allows for closed-form solutions of the rendering integral, it does not capture natural high-frequency illumination. Our work leverages the split sum approximation [11], proposed for real-time rendering of image-based global illumination to enable the learning of high-frequency environment lighting. The split sum approximation has been adopted by several inverse rendering methods [3, 14, 20]. Pre-integrated lighting has been represented as an autoencoder-based illumination network [3, 13], as a set of learnable images for different roughness levels [17, 20], and as an MLP with integrated spherical harmonic encoding as input [14]. In contrast, we propose modeling pre-integrated lighting as the output of an MLP paired with a novel regularization, which ensures the network correctly learns to represent pre-integrated lighting. An issue arising from the split sum approximation is that the pre-integration is blind to geometry

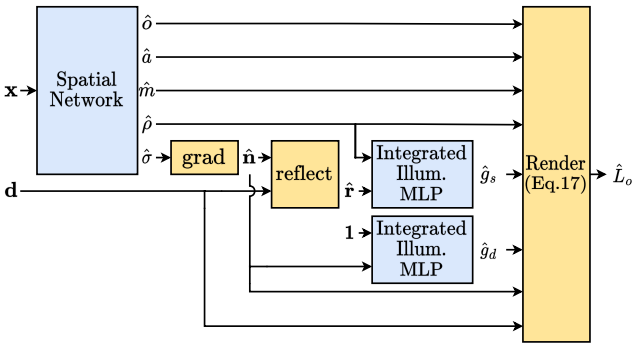

Figure 2. **Proposed Architecture.** A spatial network is used to map spatial coordinates into geometry, material properties, occlusion, and spatial features. The predicted geometry is used along with predicted roughness by the pre-integrated illumination network to predict both pre-integrated specular and diffuse terms. Finally, the pre-integrated specular and diffuse terms are combined with material properties along with an extra corrective term to produce the output radiance.

and thus does not account for the occlusion of light sources due to geometry at different locations throughout the scene. Our work tackles this issue by supervising the prediction of ambient occlusion through Monte Carlo sampling.

3. Methodology

Our method aims to extract a scene's geometry, material properties, and illumination from a set of posed images of the scene. We accomplish this by incorporating a decomposed formulation of radiance into a surface rendering pipeline. In the following sections, we begin with an overview of the surface rendering pipeline. We then detail the physically-based radiance formulation, which allows us to decompose radiance into illumination and material properties. Next, we describe our proposed MLP representation for illumination along with the additional loss term it requires. Afterward, we derive a method for estimating an occlusion factor to account for visibility within the split sum approximation. Finally, we describe additional regularization used to facilitate learning.

3.1. Overview of Neural Rendering

Neural volume rendering relies on learning two functions: $\sigma(\mathbf{x};\theta) : \mathbb{R}^3 \mapsto \mathbb{R}$ which maps a point in space $\mathbf{x}$ onto a density σ, and $\mathbf{L}_o(\mathbf{x},\omega_o;\theta) : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}^3$ that maps point $\mathbf{x}$ viewed from direction ω_o onto a radiance $\mathbf{L}_o$. The parameters θ that define the density and radiance functions are typically optimized to represent a single scene by using multiple posed views of the scene. To learn these functions, they are evaluated at multiple points along a ray $\mathbf{r}(t) = \mathbf{o} - t\omega_o$, $t \in [t_n, t_f]$, defined by the camera origin $\mathbf{o} \in \mathbb{R}^3$, pixel viewing direction ω_o, and camera near and far clipping planes t_n and t_f. A pixel color for the ray can then be obtained through volume rendering via:

$$\hat{\mathbf{C}}(\mathbf{r};\theta) = \int_{t_n}^{t_f} T(t)\,\hat{\sigma}(\mathbf{r}(t))\,\hat{L}_o(\mathbf{r}(t),\omega_o)\,\mathrm{d}t, \tag{1}$$

$$\text{where } T(t) = \exp\left(-\int_{t_n}^{t} \hat{\sigma}(\mathbf{r}(s))\,\mathrm{d}s\right). \tag{2}$$

In practice, a summation of discrete samples along the ray is used to approximate the integral. This volume rendering process allows us to supervise the learning of implicit functions L_o and σ, in a pixel-wise fashion through the reconstruction loss:

$$\mathcal{L}_{\text{rec}}(R;\theta) = \frac{1}{|R|}\sum_{\mathbf{r}\in R}\left\|\mathbf{C}(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r};\theta)\right\|_2^2, \tag{3}$$

where R is a batch of rays generated from a random subset of pixels from training images.

The learned geometry can be improved if, instead of directly predicting density σ, a signed distance field (SDF) is learned and then mapped to density. To this end, we follow the SDF formulation proposed in NeuS [33]. Learning a valid SDF requires the use of an additional Eikonal loss term $\mathcal{L}_{\text{Eik}}$. For more details, please refer to [33].

Since volume density σ depends only on a point's position in space while output radiance L_o depends on both position and viewing direction, neural rendering networks are typically split into a spatial network and a radiance network. As shown in Figure 2, we maintain the spatial network to estimate density along with additional material properties but rely on a physically-based [23] radiance estimation instead of a radiance network.

3.2. Physically-Based Rendering

Given knowledge of a scene's geometry, material properties, and illumination, it is possible to model the outgoing radiance $\mathbf{L}_o(\mathbf{x},\omega_o)$ reflected at any position $\mathbf{x}$ of an object's surface in direction ω_o by integrating over the hemisphere Ω defined by the surface's normal $\mathbf{n}$ using the reflectance equation:

$$\mathbf{L}_o = \int_{\Omega} (\mathbf{k}_d \frac{\mathbf{a}}{\pi} + \mathbf{f}_s)\mathbf{L}_i\langle\omega_i,\mathbf{n}\rangle d\omega_i, \tag{4}$$

where $\mathbf{L}_i$ is the incoming radiance, $\mathbf{a}$ is the material's diffuse albedo, and $\mathbf{k}_d$ and $\mathbf{f}_s$ are material properties dependent on the object's Bidirectional Reflectance Distribution Function (BRDF). For clarity, we omit from the notation the dependency of incoming radiance on ω_i as well as the dependency of material properties on position $\mathbf{x}$. This integral can be split into its diffuse and specular components.

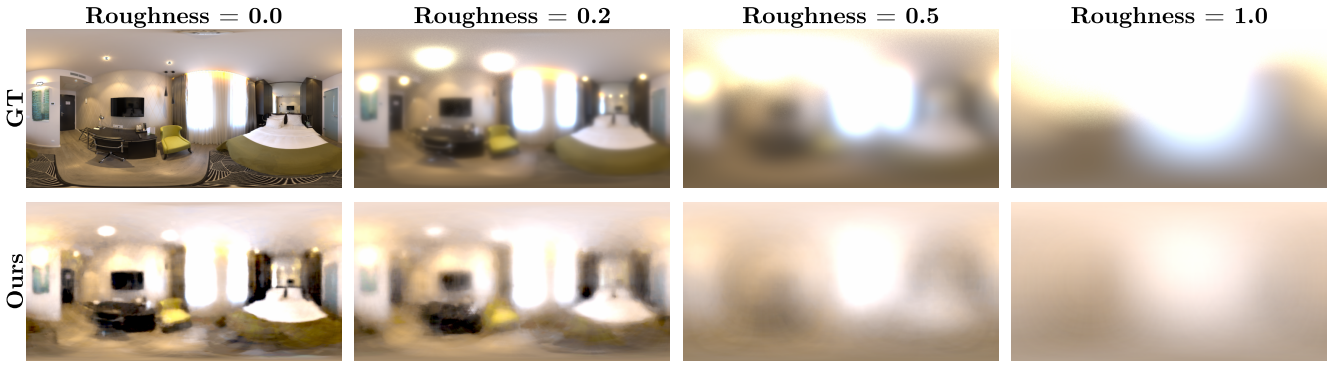

Figure 3. **Pre-Integrated environment illumination.** We visualize the pre-integrated illumination for varying roughness values along with our model's prediction for the 'toaster' scene. Our pre-integrated illumination MLP is capable of accurately approximating pre-integrated lighting across roughness values thanks to our novel Monte Carlo regularization loss.

$$\mathbf{L}_d = \mathbf{k}_d \frac{\mathbf{a}}{\pi} \int_\Omega \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i, \quad \mathbf{L}_s = \int_\Omega \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i. \tag{5}$$

Computing the specular integral for any general scene is not possible, and approximating it directly using a Monte Carlo simulation is very expensive. Thus, image-based lighting methods often employ the split sum approximation to calculate specular lighting by splitting the integral into two components: one containing the incoming light $\mathbf{L}_i$, and one that only depends on material properties independent of lighting. Modeling the BRDF using the Cook-Torrance GGX [30, 32] model leads to the following approximation for $\mathbf{L}_s$:

$$\mathbf{L}_s \approx \frac{\int_\Omega D(\omega_i, \omega_r, \rho) \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int_\Omega D(\omega_i, \omega_r, \rho) \langle \omega_i, \mathbf{n} \rangle d\omega_i} \int_\Omega \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i, \tag{6}$$

where $D(\omega_i, \omega_r, \rho)$ is the microfacet normal distribution function dependent on the direction of light reflection ω_r as well as the surface roughness ρ. The term on the right can be pre-computed since it is independent of a scene's lighting. We follow the formulation from [11] and use a two-dimensional lookup table with precomputed values F_1 and F_2. That is,

$$\begin{aligned} &\int_\Omega \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i = \mathbf{F}_r * F_1 + F_2, \\ &\mathbf{F}_r = \mathbf{F}_0 + (1 - \rho - \mathbf{F}_0) * (1 - \langle \mathbf{n}, \mathbf{v} \rangle)^5, \\ &\mathbf{F}_0 = (1 - m) * 0.04 + m * \mathbf{a}, \\ &\mathbf{k}_d = (1 - m) * (1 - \mathbf{F}_r), \end{aligned} \tag{7}$$

where m and ρ are material properties describing the metalness and roughness of a surface point respectively. As shown in Figure 2, we estimate a material's metalness $\hat{m}$, roughness $\hat{\rho}$, and albedo $\hat{\mathbf{a}}$ as additional outputs from the spatial network.

The term on the left in Equation (6) depends on the lighting and the chosen microfacet distribution function $D(\omega_i, \omega_r, \rho)$, which must be approximated whenever the environment lighting changes. In the following sections, we refer to this term as $g(\omega_r, \rho)$. For a given environment lighting, this term can be pre-integrated and is typically stored in an environmental mipmap where different mipmap levels correspond to varying values of microfacet roughness.

3.3. MLP Representation

We propose to estimate the pre-integrated lighting $g(\omega_r, \rho)$ at different roughness levels through a pre-integrated illumination MLP $\hat{g}(\hat{\omega}_r, \hat{\rho})$. That is,

$$\hat{\mathbf{L}}_s = \hat{g}(\hat{\omega}_r, \hat{\rho}) * (\hat{\mathbf{F}}_r * F_1 + F_2). \tag{8}$$

The pre-integrated lighting $g(\omega_r, \rho)$ has two special forms for the specific cases of $\rho = 0$ and $\rho = 1$.

$$g(\omega, 0) = \mathbf{L}_i(\omega), \quad g(\mathbf{n}, 1) = \frac{1}{\pi} \int_\Omega \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i, \tag{9}$$

This allows us to reuse the network $\hat{g}$ to approximate $\mathbf{L}_d$:

$$\hat{\mathbf{L}}_d = \hat{g}(\hat{\mathbf{n}}, 1) \hat{\mathbf{k}}_d \hat{\mathbf{a}}, \tag{10}$$

The predictions $\hat{g}$ should accurately represent the environment lighting at different levels of roughness. We achieve this through a loss term based on Monte Carlo estimates $\bar{g}$ of the original integral for varying roughness and reflected directions using the predicted environment map $\hat{\mathbf{L}}_i(\omega) = \hat{g}(\omega, 0)$.

$$\mathcal{L}_{\mathrm{D}}(\theta) = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \|\hat{g}(s) - \bar{g}(s)\|_2^2,$$
$$\bar{g}(s) = \frac{\sum_{\omega_i\in\Omega} D(\omega_i,\omega_s,\rho_s)\hat{g}(\omega_i,0)\langle\omega_i,\omega_s\rangle}{\sum_{\omega_i\in\Omega} D(\omega_i,\omega_s,\rho_s)\langle\omega_i,\omega_s\rangle}, \tag{11}$$

where the set $\mathcal{S}$ consists of paired samples of directions ω_s taken uniformly on a sphere, and roughness samples ρ_s with half the samples taken uniformly in the range $[0, 1]$ and the other half fixed to 1 to ensure correct learning of diffuse lighting. The set Ω of light direction samples is also taken uniformly on a sphere. While a different sampling could lead to reduced variance, we utilize uniform spherical sampling for ω_i to be more computationally efficient. Uniform spherical sampling allows us to share light samples across the batch of predictions, thus reducing the number of evaluation calls to the light function $\hat{g}(\omega, 0)$. We visualize both g and $\hat{g}$ in Figure 3 for a specific scene.

3.4. Occlusion Factors

The split sum approximation does not consider the occlusion of light sources due to geometry. To incorporate occlusions, incoming light $\mathbf{L}_i$ would need to be multiplied by a binary visibility function V_i as follows:

$$\mathbf{L}_d^V = \mathbf{k}_d\frac{\mathbf{a}}{\pi}\int_\Omega \mathbf{L}_i V_i\langle\omega_i,\mathbf{n}\rangle d\omega_i, \tag{12}$$

with V_i taking a value of 1 when there are no occlusions and 0 when incoming light is occluded by geometry. The integral can be written as an occlusion factor $\mathbf{o}_d(\mathbf{x})$ multiplying the split sum diffuse light term from Equation (5):

$$\int_\Omega \mathbf{L}_i V_i\langle\omega_\mathbf{i},\mathbf{n}\rangle d\omega_i = \frac{\int_\Omega \mathbf{L}_i V_i\langle\omega_i,\mathbf{n}\rangle d\omega_i}{\int_\Omega \mathbf{L}_i\langle\omega_i,\mathbf{n}\rangle d\omega_i}\int_\Omega \mathbf{L}_i\langle\omega_i,\mathbf{n}\rangle d\omega_i,$$
$$\mathbf{L}_d^V = \mathbf{o}_d(\mathbf{x})\mathbf{L}_d, \quad \mathbf{o}_d(\mathbf{x}) = \frac{\int_\Omega \mathbf{L}_i V_i\langle\omega_i,\mathbf{n}\rangle d\omega_i}{\int_\Omega \mathbf{L}_i\langle\omega_i,\mathbf{n}\rangle d\omega_i} \tag{13}$$

We propose learning the occlusion factor $\mathbf{o}_d(\mathbf{x})$ with an MLP. The learnt occlusion term $\hat{\mathbf{o}}_d(\mathbf{x})$ is then supervised by Monte Carlo estimates $\bar{\mathbf{o}}_d(\mathbf{x})$ using the predicted geometry.

$$\bar{\mathbf{o}}_d(\mathbf{x}) = \frac{\sum_{\omega_i\in\Omega}\mathbf{L}_i V_i}{\sum_{\omega_i\in\Omega}\mathbf{L}_i}, \tag{14}$$

with ω_i taken from a cos-weighted sampling of the hemisphere around the normal at $\mathbf{x}$. A similar derivation can be followed for the specular occlusion term leading to the following Monte Carlo estimate $\bar{\mathbf{o}}_s(\mathbf{x})$:

$$\bar{\mathbf{o}}_s(\mathbf{x}) = \frac{\sum_{\omega_i\in\Omega}\mathbf{L}_i V_i\langle\omega_i,\mathbf{n}\rangle}{\sum_{\omega_i\in\Omega}\mathbf{L}_i\langle\omega_i,\mathbf{n}\rangle}, \tag{15}$$

with ω_i sampled from the GGX distribution centered around the normal at $\mathbf{x}$. Given the Monte Carlo estimates $\bar{\mathbf{o}}_d(\mathbf{x})$ and $\bar{\mathbf{o}}_s(\mathbf{x})$, we supervise the predicted occlusion terms $\hat{\mathbf{o}}_d(\mathbf{x})$ and $\hat{\mathbf{o}}_s(\mathbf{x})$ as follows:

$$\mathcal{L}_{\mathrm{o}}(\theta) = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} w\,\|\hat{\mathbf{o}}(\mathbf{x}) - \bar{\mathbf{o}}(\mathbf{x})\|_2^2, \tag{16}$$

where the sample set $\mathcal{X}$ is a random subset of the points sampled for volume rendering, and the weights w are the corresponding normalized volume rendering weights. Weighting the loss function by the volume rendering weights is required so that the occlusion prediction focuses only on learning surface points.

The output radiance at each point in space is thus calculated as follows:

$$\hat{\mathbf{L}}_o = \gamma(\hat{\mathbf{o}}_d * \hat{\mathbf{L}}_d + \hat{\mathbf{o}}_s * \hat{\mathbf{L}}_s), \tag{17}$$

where γ is a function mapping the predicted output radiance $\hat{\mathbf{L}}_o$ from linear to SRGB space.

3.5. Material Regularization

To better learn material properties, we introduce a soft regularizer to reduce the prediction of metallic materials. This encourages the model to prefer explaining outgoing radiance through albedo and roughness whilst still allowing the prediction of metallic materials. We implement this regularization as a weighted L_2 loss with the same weighting as for the occlusion loss in Equation (16). That is,

$$\mathcal{L}_{\mathrm{m}}(\theta) = \frac{1}{|\mathcal{X}|}\sum_{\mathbf{x}\in\mathcal{X}} w\,\|\hat{m}(\mathbf{x})|_2^2. \tag{18}$$

4. Experiments

4.1. Baselines

We compare against Nerfactor [41], NVDiffRec [20], NVDiffRecMC [8], NMF [16], and NeRO [14]. Due to the differing evaluation methodologies among these works, we train all baseline methods following publicly released code and report metrics as detailed in the following sections.

4.2. Experimental setup

Datasets. We report results using the NeRFactor [41] dataset along with extended versions of the NeRF Blender [18] (Blender) and the RefNeRF Shiny

	PSNR ↑					SSIM ↑					LPIPS ↓				
	avg.	**drums**	**ficus**	**hotdog**	**lego**	**avg.**	**drums**	**ficus**	**hotdog**	**lego**	**avg.**	**drums**	**ficus**	**hotdog**	**lego**
NerFactor	23.66	20.01	23.71	26.15	24.77	0.895	0.879	0.932	0.914	0.854	0.120	0.130	0.090	0.118	0.141
NVDiffRec	21.88	20.72	20.09	24.64	22.09	0.880	0.890	0.907	0.892	0.831	0.111	0.097	0.085	0.124	0.137
NVDiffRecMC	24.06	21.56	21.38	**29.05**	24.24	0.902	0.899	0.910	0.938	0.862	0.099	0.094	0.079	0.089	0.134
NMF	22.23	21.54	21.36	22.47	23.58	0.895	0.906	0.934	0.876	0.863	0.093	0.075	0.063	0.120	0.116
NERO	23.68	20.73	23.58	25.28	25.14	0.907	0.900	0.936	0.908	0.884	0.093	0.110	0.062	0.093	0.108
Ours	**27.31**	**24.72**	**27.45**	29.04	**28.02**	**0.941**	**0.935**	**0.964**	**0.947**	**0.920**	**0.061**	**0.058**	**0.041**	**0.069**	**0.075**

Table 1. **NeRFactor Relighting Metrics.** We evaluate the relighting quality of our method against the baselines using 20 test images and 8 low-frequency illumination maps from the NeRFactor dataset. Images are scaled by a per-channel factor before computing metrics. Our method outperforms the baselines across all reconstruction metrics for all but one scene.

	Blender			Shiny Blender		
	PSNR ↑	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
NVDiffRec	20.11	0.857	0.138	21.39	0.848	0.177
NVDiffRecMC	22.50	0.884	0.136	24.60	**0.911**	0.151
NMF	21.21	0.881	0.118	24.20	0.908	**0.136**
Ours	**22.73**	**0.906**	**0.106**	**24.96**	0.904	0.144

Table 2. **Blender and Shiny Blender Relighting Metrics.** We report the average relighting reconstruction metrics across all scenes for our extended Blender and Shiny Blender datasets. Metrics are computed as the average of 20 test views across 7 high-frequency illumination conditions. Images are scaled by a per-channel factor before computing metrics. Our method outperforms the baselines across all metrics for the Blender dataset and has a higher PSNR for the Shiny Blender dataset.

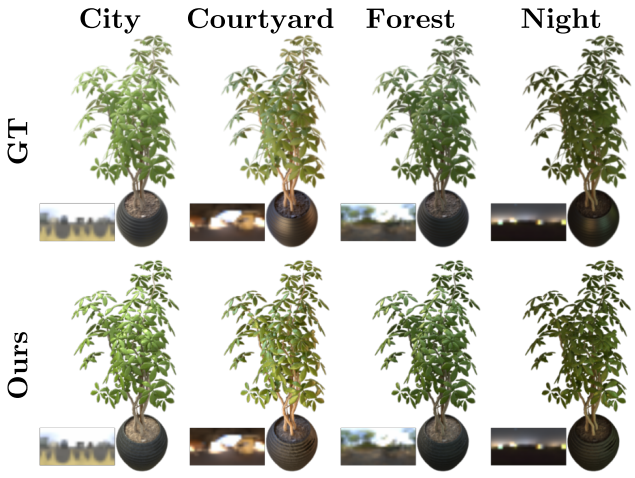

Figure 4. **Qualitative Relighting Results.** We render the predicted mesh and material properties from the 'ficus' scene using Blender. Four different low-frequency environment maps from the NeRFactor dataset are visualized.

Blender [31] (Shiny Blender) datasets. The NeRFactor dataset consists of four synthetic scenes, where test images are rendered under eight different low-frequency lighting conditions. The Blender dataset consists of eight synthetic scenes representing a mix of glossy, specular, and Lambertian objects, while the Shiny Blender dataset consists of six highly reflective synthetic scenes. To showcase the ability of our model to estimate high-frequency environment lighting, we extend the Blender and Shiny Blender datasets by rendering all objects under seven novel high-frequency lighting conditions. All models are trained using 100 posed images, and evaluated on 20 test images consisting of novel views for each lighting condition.

Implementation Details. We utilize an efficient implementation of NeuS [7] as our surface rendering pipeline. We train our models for $20,000$ steps using a warmup learning rate scheduler for the first 500 steps followed by an exponential decay scheduler. After every 2000 steps, we estimate the current geometry by using marching cubes [15] to extract the isosurface at SDF level-set 0. The estimated geometry is used with 64 samples for the Monte Carlo estimation of occlusion factors. We use a random subset of 10% of the points from volume rendering to supervise the occlusion network to reduce time and memory requirements. 8129 light samples are used for computing illumination loss Monte Carlo estimates. The final loss is calculated as a linear combination of the proposed losses, with the following coefficients: $\lambda_{\text{rec}} = 10.0$, $\lambda_D = 10.0$, $\lambda_o = 0.01$, $\lambda_{\text{Eik}} = 0.1$, and $\lambda_m = 0.001$. We run all experiments on a single A100 GPU for a total training time of $\sim$1 hour.

4.3. Relighting

We extract geometry from our model in the form of a triangular mesh by using marching cubes [15]. At each predicted mesh vertex, we estimate material properties in the form of an albedo, metalness, and roughness. We then render the predicted geometry using Blender's [5] physically based shader. Material properties across faces are obtained by interpolating the predicted vertex material properties. For baselines where explicit meshes and material properties are extracted, we utilize the same Blender rendering pipeline to compute relighting metrics. Otherwise, predictions are rendered using the provided relighting methodology. Before evaluating metrics, a per-channel scaling factor is computed for each scene to compensate for the albedo-lighting ambiguity. We evaluate the predicted scenes for the NeRFactor, Blender, and Shiny Blender datasets and report the average Peak Signal-to-Noise Ratio (PSNR), Structural Similarity

	PSNR ↑					SSIM ↑					LPIPS ↓				
	avg.	**drums**	**ficus**	**hotdog**	**lego**	**avg.**	**drums**	**ficus**	**hotdog**	**lego**	**avg.**	**drums**	**ficus**	**hotdog**	**lego**
NerFactor	23.53	20.75	22.05	27.75	**23.58**	0.910	0.878	0.923	0.937	**0.903**	0.109	0.132	0.098	**0.093**	**0.112**
NVDiffRec	22.96	19.66	22.64	28.35	21.20	0.898	0.880	0.924	0.944	0.842	0.125	0.120	0.099	0.108	0.172
NVDiffRecMC	23.83	20.40	22.14	**29.61**	23.17	0.917	0.895	0.921	**0.950**	0.903	0.115	0.116	0.106	0.101	0.136
NeRO	21.18	20.00	20.42	23.27	21.01	0.871	0.880	0.900	0.883	0.821	0.141	0.138	0.125	0.125	0.176
Ours	**25.29**	**23.73**	**29.41**	24.68	23.33	**0.924**	**0.922**	**0.974**	0.929	0.870	**0.108**	**0.099**	**0.051**	0.121	0.160

Table 3. **Quantitative Albedo Metrics.** We report the albedo reconstruction quality of our method compared to the baselines using the NeRFactor dataset. Albedo is scaled by a per-channel factor to minimize error. On average, we outperform all baselines across all metrics.

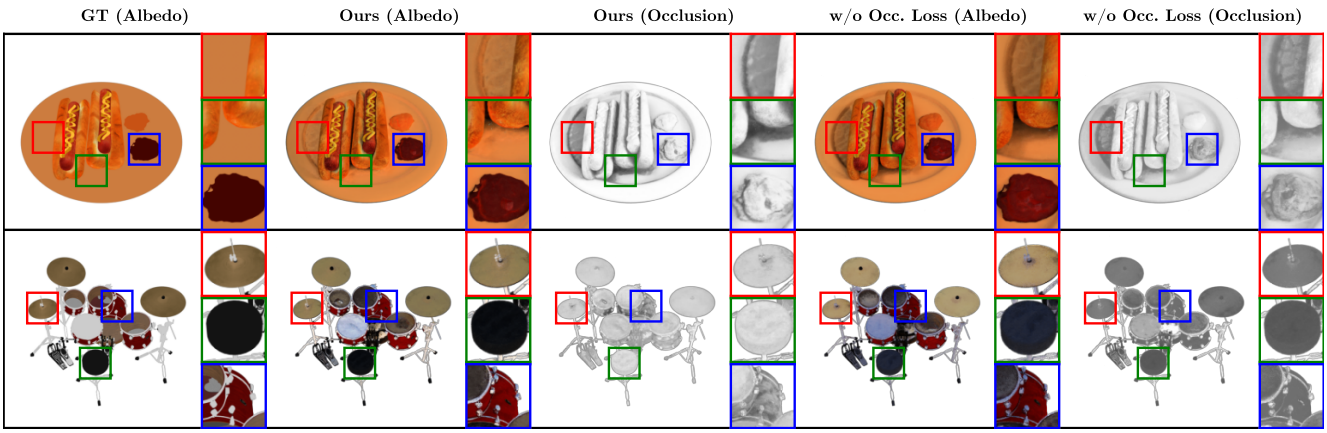

Figure 5. **Occlusion Loss Visualization.** We visualize the albedo and occlusion predicted by our method with and without the proposed occlusion regularization loss. When no regularization is used, we observe that the occlusion prediction fails at disentangling shadows from the albedo. Additionally, darker materials might wind up with lighter albedos due to occlusion overcompensation.

	Relighting			Albedo		
	PSNR ↑	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
Ours	27.31	0.941	0.061	25.29	**0.924**	**0.108**
Ours (w/o Occ. Avg.)	27.25	0.941	0.061	24.73	0.919	0.109
Ours (w/o Occ. Loss)	26.40	0.936	0.064	22.13	0.900	0.122
Ours (w/o Met. Reg.)	26.56	0.936	0.066	23.92	0.916	0.123
Ours (Mipmap)	**27.81**	**0.944**	**0.058**	**25.43**	0.924	0.114

Table 4. **NeRFactor Ablation Results.** We report relighting and ablation metrics for different variations of our methodology on the NeRFactor dataset. While employing a mipmap representation for illumination provides improved quality for both relighting and albedo, it comes at a cost of ∼36% higher training time. All other proposed components of our method improve quality.

Index Measure (SSIM) [35], and Learned Perceptual Image Patch Similarity (LPIPS) [40] in Table 1 and Table 2. Metrics are reported as an average across 20 test images and across all illumination maps for each dataset. This metric gives an aggregated performance measure for geometry and material property estimation. While it does not measure performance for estimating illumination, an accurate illumination estimation is essential for recovering material properties. It can be observed that our method outperforms all baselines by a significant margin. Additionally, we provide renderings of our method's predictions for the NeRFactor dataset's ficus scene under four different illumination conditions in Figure 4.

4.4. Albedo

In addition to overall relighting quality, we evaluate the ability of our method to recover albedo. We report reconstruction metrics on the predicted albedo in Table 3. As with the relighting metrics, we apply a per-scaling factor to the albedo predictions before computing reconstruction metrics. Metrics are reported as an average across all 20 test images for each scene in the NeRFactor dataset. We exclude results from NMF [16] since the albedo in their lighting formulation is not comparable to that of the other methods. Thanks to our proposed occlusion factor, our method is on average better able to reconstruct albedo.

5. Discussions

5.1. Ablations

Occlusion loss. We visualize the effects of the proposed occlusion loss in Figure 5. Learning an occlusion factor without supervision leads to errors in the albedo predictions due to the inability to disentangle shadows from object color. By explicitly supervising an occlusion factor we observe better albedo color predictions such as visualized in the blue box in the hotdog example, and all boxes in the drums example. Additionally, shadows are better disentangled from albedo as observed in the red and green boxes for the hot-

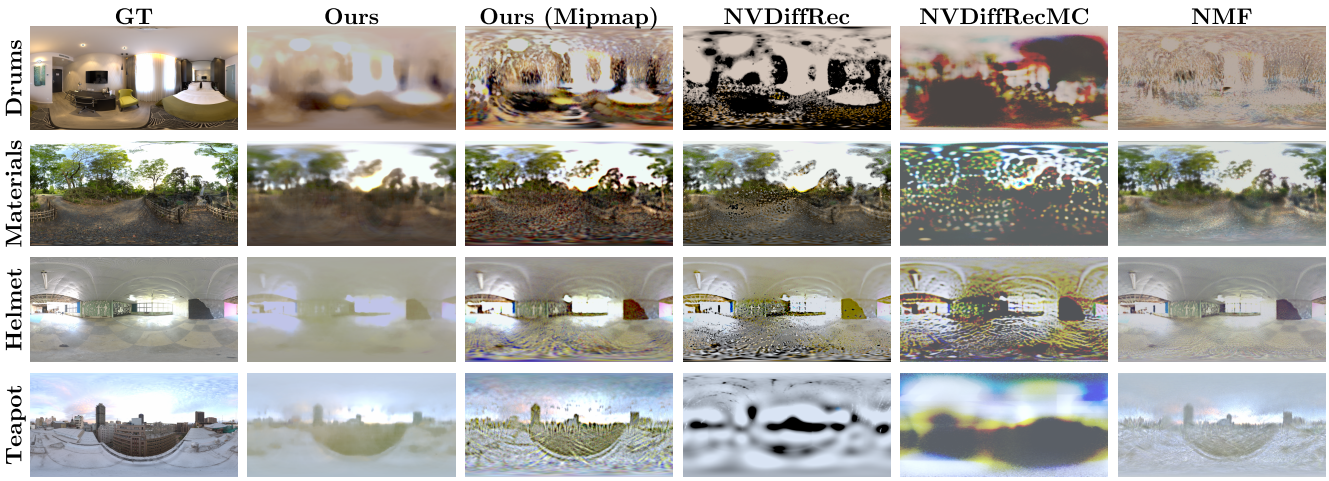

Figure 6. **Blender and Shiny Blender Illumination Visualizations.** We visualize the predicted illumination for our method, our method using mipmap illumination, and baselines for two scenes in the Blender dataset and two scenes in the Shiny Blender dataset. Illumination is scaled by a per-channel factor. Our proposed illumination inherits smoothness from the MLP representation but is still able to capture high-quality details such as trees and buildings.

dog example. Quantitatively, we measure the importance of adding the occlusion loss to our model in Table 4, where it improves both relighting and albedo reconstruction.

Material Regularization. We measure the effect of the material regularization in Table 4. By penalizing metalness prediction, our model tries to explain output radiance through roughness and albedo. However, as visualized in Figure 1, the loss coefficient used is small enough to still allow our model to correctly predict metalness when required.

Occlusion Averaging. The occlusion factor we derive consists of a per-channel factor that depends on estimated lighting. However, since both are being learned jointly, we observe that this can be a noisy process. We find in Table 4 that relighting and albedo reconstruction both improve when we supervise the occlusion factors $\hat{o}_d$ and $\hat{o}_s$ with their per-channel averages instead. Assuming that all channels of the occlusion factor are equal is equivalent to assuming that all color channels of the lighting are equal, which slightly reduces the noise during training and uses fewer parameters.

MLP vs. Mipmap. We compare the effects of using a mipmap representation for illumination against our proposed regularized MLP. As reported in Table 4, the mipmap representation leads to slight improvements in both relighting and albedo reconstruction. However, these improvements come at a cost of efficiency, since the average training time for our method is only 47 min., which increases to 64 min. with the mipmap representation - a 36% increase in training time. As visualized in Figure 6, our method produces smoother representations with less noise thanks to the MLP representation. However, it is still capable of capturing fine details such as trees in the 'materials' scene and buildings in the 'teapot' scene. We expect further improvements in the rapidly advancing field of neural rendering will translate to better MLP representations and benefit our method.

6. Conclusion and Limitations

In conclusion, we present a novel and efficient method for inverse rendering based on neural surface rendering and the split sum approximation for image-based lighting. Owing to our proposed integrated illumination MLP, we can jointly estimate geometry, lighting, and material properties in under one hour using a single NVIDIA A100 GPU. Additionally, we propose a way of supervising an occlusion factor for diffuse and specular lighting such that self-occlusions are accounted for with the split sum approximation. Altogether, our method is capable of producing high-quality estimates of geometry, lighting, and material properties as measured by rendering objects under unseen views and lighting conditions.

However, due to the highly complex problem that inverse rendering presents, there are some limitations to our method. The major assumptions we rely on come from using image-based lighting and the split sum approximation. Image-based lighting assumes that light sources are located infinitely far away from the scene, leading to errors when this assumption is violated. While we have tackled the problem of missing self-occlusions within the split sum approximation, we ignore the lack of indirect illumination. Additionally, we only consider the reflection of light and are unable to model transmission and subsurface scattering effects. We hope future works will be able to tackle some of these limitations.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 1, 2

[2] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2

[3] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P.A. Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

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

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 6

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1

[7] Yuan-Chen Guo. Instant neural surface reconstruction, 2022. https://github.com/bennyguo/instant-nsr-pl. 6

[8] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. *arXiv:2206.03380*, 2022. 5

[9] Yuchen Jiang, Shen Yin, Kuan Li, Hao Luo, and Okyay Kaynak. Industrial applications of digital twins. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379:20200360, 2021. 1

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

[11] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 4(3):1, 2013. 2, 4

[12] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 12

[13] Ruofan Liang, Huiting Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. Envidr: Implicit differentiable renderer with neural environment lighting. *arXiv preprint arXiv:2303.13022*, 2023. 2

[14] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. In *SIGGRAPH*, 2023. 2, 5

[15] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 6

[16] Alexander Mai, Dor Verbin, Falko Kuester, and Sara Fridovich-Keil. Neural microfacet fields for inverse rendering, 2023. 1, 2, 5, 7

[17] Shi Mao, Chenming Wu, Zhelun Shen, and Liangjun Zhang. Neus-pir: Learning relightable neural surface using pre-integrated rendering. *CoRR*, abs/2306.07632, 2023. 2

[18] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 5, 12

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

[20] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8280–8290, 2022. 2, 5

[21] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, 126(9):902–919, 2018. 1

[22] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[23] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016. 3

[24] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. 1

[25] Sara Rojas, Jesus Zarzar, Juan C. Pérez, Artsiom Sanakoyeu, Ali Thabet, Albert Pumarola, and Bernard Ghanem. Rerend: Real-time rendering of nerfs across devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3632–3641, 2023. 2

[26] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. 1

[27] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 1, 2

[28] Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural 3d reconstruction in the wild. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 2

[29] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. *arXiv*, 2022. 1

[30] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces∗. *J. Opt. Soc. Am.*, 57(9): 1105–1114, 1967. 4

[31] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 1, 2, 6

[32] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, page 195–206, Goslar, DEU, 2007. Eurographics Association. 4

[33] 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. *arXiv preprint arXiv:2106.10689*, 2021. 2, 3

[34] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. *Advances in Neural Information Processing Systems*, 35:1966–1978, 2022. 2

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

[36] Haoqian Wu, Zhipeng Hu, Lincheng Li, Yongqiang Zhang, Changjie Fan, and Xin Yu. Nefii: Inverse rendering for reflectance decomposition with near-field indirect illumination. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4295–4304, 2023. 2

[37] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2

[38] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

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

[40] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, Los Alamitos, CA, USA, 2018. IEEE Computer Society. 7

[41] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 2021. 2, 5

[42] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *CVPR*, 2022. 2

SplitNeRF: Split Sum Approximation Neural Field for Joint Geometry, Illumination, and Material Estimation

Supplementary Material

7. Appendix

7.1. Derivation of Illumination Loss

In this section, we go through the derivation for the Monte Carlo approximation of pre-integrated illumination $\bar{g}$ used in Equation (11). We first split the specular light integral into two terms:

$$\mathbf{L}_s = \frac{\int\limits_{\Omega} \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int\limits_{\Omega} \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i} \int\limits_{\Omega} \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i. \tag{19}$$

As mentioned in the main paper, the term on the right can be precomputed so we focus on calculating an approximation for the term on the left.

$$g(\omega_r, \rho) \approx \frac{\int\limits_{\Omega} \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int\limits_{\Omega} \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i}. \tag{20}$$

This term requires us to make two approximations to the Cook-Torrance BRDF $\mathbf{f}_s$

$$\mathbf{f}_s = \frac{DFG}{4\langle \omega_\mathbf{o}, \mathbf{n} \rangle \langle \omega_\mathbf{i}, \mathbf{n} \rangle}, \tag{21}$$

to be able to approximate g as blurred environment maps as per the split sum approximation. The first approximation on the BRDF consists of assuming the multiplication between fresnel and geometric shadowing terms is approximately equal to the dot product between the normal and viewing directions: $FG \approx \langle \omega_i, \mathbf{n} \rangle$. Thus, we have that

$$\mathbf{f}_s \approx \frac{D}{4\langle \omega_o, \mathbf{n} \rangle}, \quad g(\omega_r, \rho) \approx \frac{\int\limits_{\Omega} D \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int\limits_{\Omega} D \langle \omega_i, \mathbf{n} \rangle d\omega_i}, \tag{22}$$

as shown in Equation (6). We use the GGX (Trowbridge-Reitz) microfacet distribution function for D:

$$D(\omega_i, \omega_o, \mathbf{n}, \rho) = \frac{\rho^2}{\pi(\langle \mathbf{h}, \mathbf{n} \rangle^2 (\rho^2 - 1) + 1)^2}, \tag{23}$$

where $\mathbf{h}$ is the half vector between ω_i and ω_o. The second approximation assumes the normal and viewing directions to be equal to the reflection direction. That is, $\mathbf{n} \approx \omega_r$ and $\omega_o \approx \omega_r$. This leaves us with the following simplified D:

$$D(\omega_i, \omega_r, \rho) \approx \frac{\rho^2}{\pi(\frac{1+\langle \omega_i, \omega_r \rangle}{2}(\rho^2 - 1) + 1)^2}, \tag{24}$$

which now does not depend on either the normal or viewing directions. Approximating both integrals with Monte Carlo sampling and taking the same number of samples, we arrive at the expression in Equation (11):

$$\bar{g}(\omega_r, \rho) = \frac{\sum\limits_{\Omega} D(\omega_i, \omega_r, \rho) \mathbf{L}_i \langle \omega_i, \omega_r \rangle d\omega_i}{\sum\limits_{\Omega} D(\omega_i, \omega_r, \rho) \langle \omega_i, \omega_r \rangle d\omega_i}. \tag{25}$$

7.2. Derivation of Occlusion Factor Approximation

We now go over the derivation of the occlusion factor Monte Carlo approximation. As shown in Equation (13), we aim to approximate the occlusion factors $\mathbf{o}_d(\mathbf{x})$ and $\mathbf{o}_s(\mathbf{x})$

$$\mathbf{o}_d(\mathbf{x}) = \frac{\int\limits_{\Omega} \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int\limits_{\Omega} \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}, \tag{26}$$

$$\mathbf{o}_s(\mathbf{x}) = \frac{\int\limits_{\Omega} D \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int\limits_{\Omega} D \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}, \tag{27}$$

in order to supervise an occlusion factor network through Monte Carlo sampling. As shown in the main manuscript, we approximate o_d with Monte Carlo sampling by taking the same number of samples for both integrals:

$$\bar{\mathbf{o}}_d(\mathbf{x}) = \frac{\sum\limits_{\omega_i \in \Omega} \mathbf{L}_i V_i}{\sum\limits_{\omega_i \in \Omega} \mathbf{L}_i}, \tag{28}$$

with ω_i taken from a cos-weighted sampling of the hemisphere around the normal $\mathbf{n}$ at location $\mathbf{x}$. The probability density function sampled is given by:

$$\text{pdf}(\omega_i) = \frac{\cos(\theta)}{\pi}, \tag{29}$$

where θ is the angle between ω_i and the normal. This cos-weighted sampling aids in reducing variance by eliminating the dot product factor from the estimation. A similar derivation can be followed for the specular occlusion term leading to the following Monte Carlo estimate $\bar{\mathbf{o}}_s(\mathbf{x})$:

$$\bar{\mathbf{o}}_s(\mathbf{x}) = \frac{\sum\limits_{\omega_i \in \Omega} \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle}{\sum\limits_{\omega_i \in \Omega} \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle}, \tag{30}$$

where ω_i is now obtained by sampling the GGX distribution in order to reduce variance by eliminating the factor D from both integrals. The probability density function sampled in this case is the following:

$$\text{pdf}(\omega_i) = \frac{D(\omega_i, \omega_r, \rho)}{4}, \tag{31}$$

which relies on the second approximation used in the previous section.

7.3. Network Implementation Details

We implement the spatial network using the progressive hash grid encoding from [12]. The hash grid consists of 16 levels with 2 features per level and a hashmap size of 2^{19} entries. The base grid spatial resolution is 32 voxels, increasing by $\sim$1.32 each level. An MLP with a single 64-channel hidden layer is used to produce spatial features with 13 channels along with the SDF predictions. Spatial features are then input to an MLP with two hidden layers of 256 channels each and ReLU activations to produce material property (metalness, roughness, and albedo) predictions. A separate but identical MLP is used to produce occlusion factor predictions. A sigmoid is used to map the MLP outputs to the occlusion factor and material properties' ranges of $[0, 1]$. The illumination network consists of an MLP with five hidden layers with 256 channels each and ReLU activations. Both the direction and roughness vectors used as input to the illumination network are first positionally encoded as proposed in [18], using 10 frequencies for the directional input and 5 for the roughness input. A softplus function is used to map the illumination network's output to the range $(0, \text{inf})$.

7.4. Blender and Shiny Blender Relighting Results

We report per-scene metrics for relighting using the Blender dataset in Tables 5 to 7, and using the Shiny Blender dataset in Tables 8 to 10. Additionally, we present qualitative results of our method visualizing the learnt illumination, material properties (metalness, roughness, and albedo), geometry, and relit renderings from our method's predictions for the Blender dataset in Figures 7 to 14 and for the Shiny Blender dataset in Figures 15 to 19.

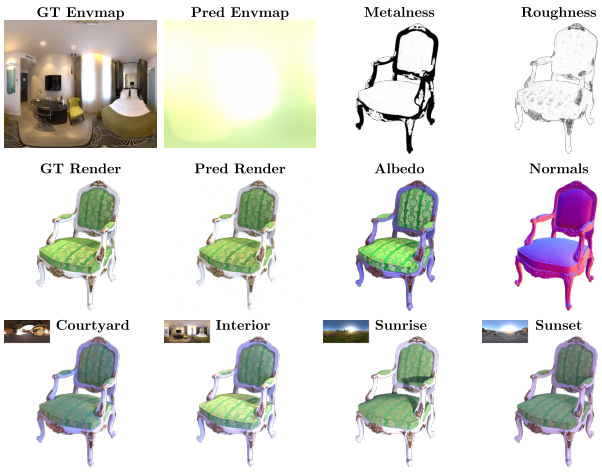

Figure 7. **Qualitative results on the Blender 'chair' scene.**

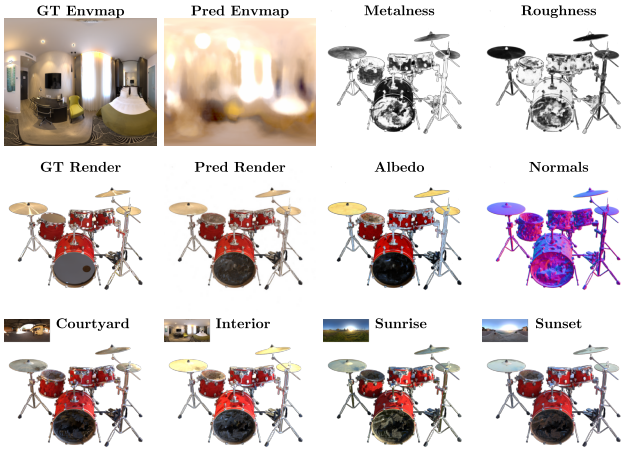

Figure 8. **Qualitative results on the Blender 'drums' scene.**

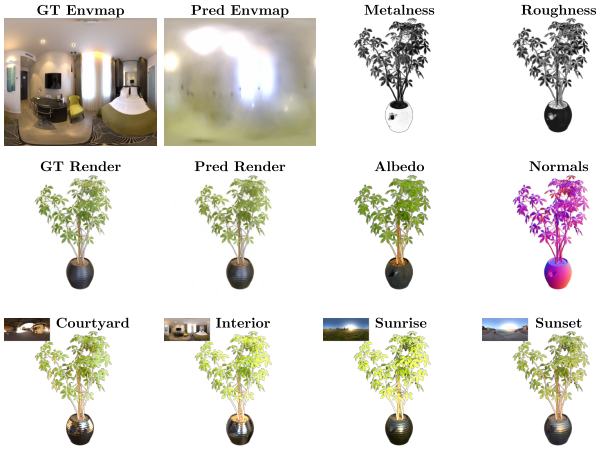

Figure 9. **Qualitative results on the Blender 'ficus' scene.**

	PSNR ↑								
	avg.	**chair**	**drums**	**ficus**	**hotdog**	**lego**	**materials**	**mic**	**ship**
NVDiffRec	20.11	21.70	19.41	20.32	21.17	21.84	21.26	18.48	16.69
NVDiffRecMC	22.50	24.64	20.57	21.27	**26.37**	**24.77**	24.57	**18.95**	**18.91**
NMF	21.21	22.53	21.38	22.62	20.52	22.05	24.87	18.34	17.41
Ours	**22.73**	**25.00**	**22.62**	**26.40**	20.94	23.88	**25.38**	18.75	18.88

Table 5. **Blender per-scene PSNR.**

	SSIM ↑								
	avg.	**chair**	**drums**	**ficus**	**hotdog**	**lego**	**materials**	**mic**	**ship**
NVDiffRec	0.857	0.886	0.852	0.902	0.894	0.834	0.866	0.927	0.695
NVDiffRecMC	0.884	0.918	0.877	0.902	**0.930**	0.865	0.904	0.924	0.750
NMF	0.881	0.908	0.890	0.934	0.890	0.863	0.913	0.926	0.722
Ours	**0.906**	**0.937**	**0.908**	**0.952**	0.914	**0.901**	**0.930**	**0.937**	**0.769**

Table 6. **Blender per-scene SSIM.**

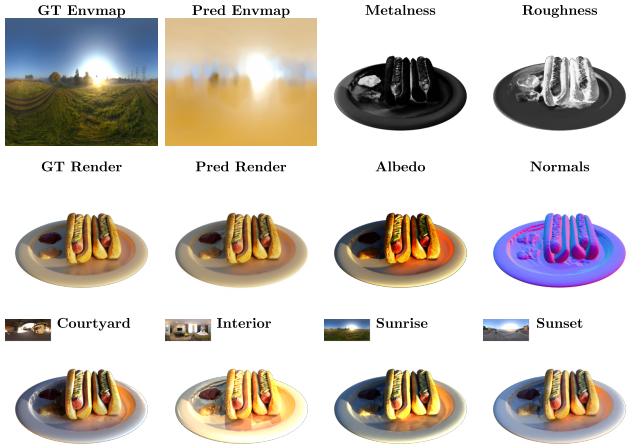

Figure 10. **Qualitative results on the Blender 'hotdog' scene.**

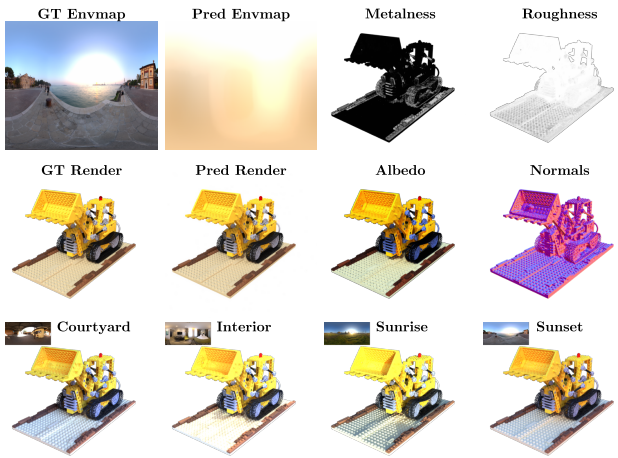

Figure 11. **Qualitative results on the Blender 'lego' scene.**

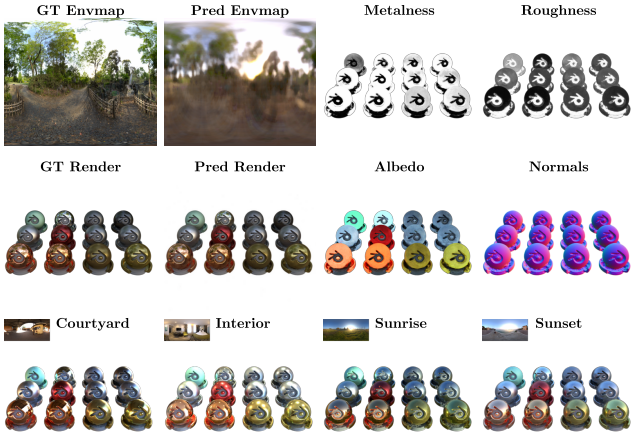

Figure 12. **Qualitative results on the Blender 'materials' scene.**

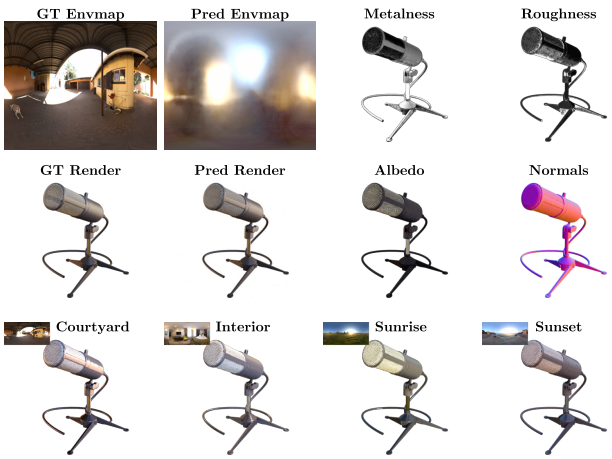

Figure 13. **Qualitative results on the Blender 'mic' scene.**

	LPIPS ↓								
	avg.	**chair**	**drums**	**ficus**	**hotdog**	**lego**	**materials**	**mic**	**ship**
NVDiffRec	0.138	0.099	0.134	0.083	0.141	0.141	0.132	0.092	0.283
NVDiffRecMC	0.136	0.084	0.135	0.093	0.118	0.147	0.112	0.097	0.306
NMF	0.118	0.093	0.103	0.066	0.135	0.108	0.081	0.087	0.271
Ours	**0.106**	**0.065**	**0.096**	**0.050**	**0.116**	**0.097**	**0.075**	**0.077**	**0.269**

Table 7. **Blender per-scene LPIPS.**

	PSNR ↑					
	avg.	**car**	**coffee**	**helmet**	**teapot**	**toaster**
NVDiffRec	21.39	24.85	18.29	19.07	29.39	15.36
NVDiffRecMC	24.60	24.25	**22.93**	22.86	32.70	20.25
NMF	24.20	24.58	17.88	**27.48**	30.66	**20.41**
Ours	**24.96**	**26.86**	18.70	21.51	**38.13**	19.62

Table 8. **Shiny Blender per-scene PSNR.**

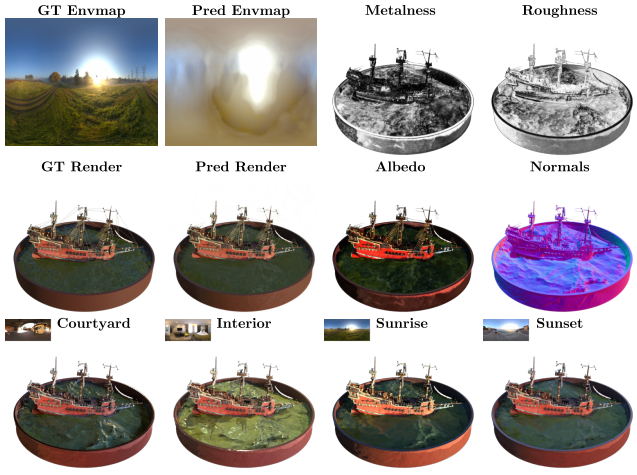

Figure 14. **Qualitative results on the Blender 'ship' scene.**

Figure 15. **Qualitative results on the Shiny Blender 'car' scene.**

Figure 16. **Qualitative results on the Shiny Blender 'coffee' scene.**

	SSIM ↑					
	avg.	**car**	**coffee**	**helmet**	**teapot**	**toaster**
NVDiffRec	0.848	0.913	0.799	0.848	0.972	0.705
NVDiffRecMC	**0.911**	0.917	**0.921**	0.908	0.983	0.827
NMF	0.908	0.917	0.843	**0.946**	0.982	**0.849**
Ours	0.904	**0.942**	0.883	0.877	**0.993**	0.827

Table 9. **Shiny Blender per-scene SSIM.**

	LPIPS ↓					
	avg.	**car**	**coffee**	**helmet**	**teapot**	**toaster**
NVDiffRec	0.177	0.102	0.237	0.209	0.046	0.290
NVDiffRecMC	0.151	0.101	**0.195**	0.182	0.039	0.241
NMF	**0.136**	0.092	0.208	**0.142**	0.032	**0.206**
Ours	0.144	**0.072**	0.204	0.195	**0.017**	0.234

Table 10. **Shiny Blender per-scene LPIPS.**

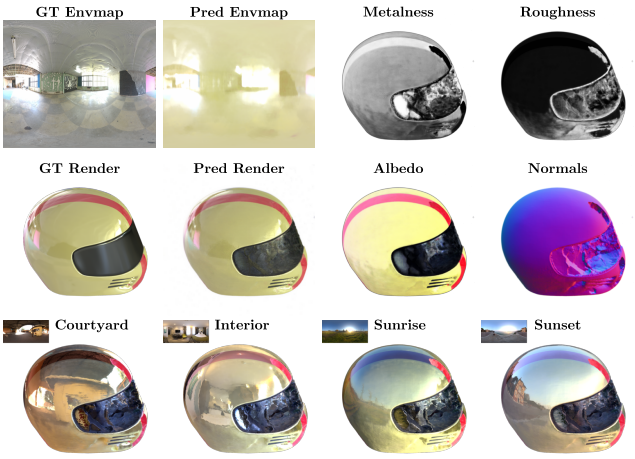

Figure 17. **Qualitative results on the Shiny Blender 'helmet' scene.**

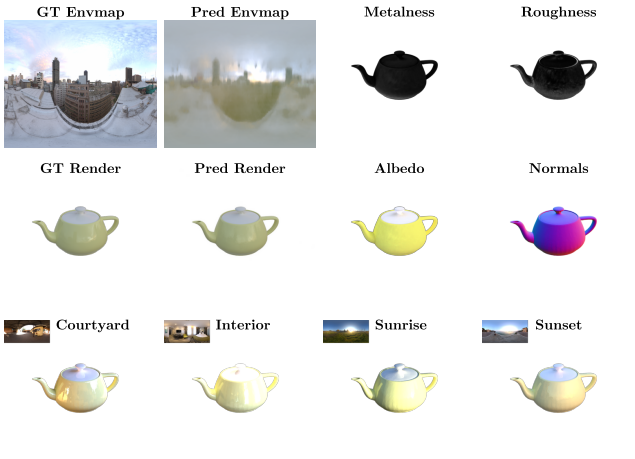

Figure 18. **Qualitative results on the Shiny Blender 'teapot' scene.**

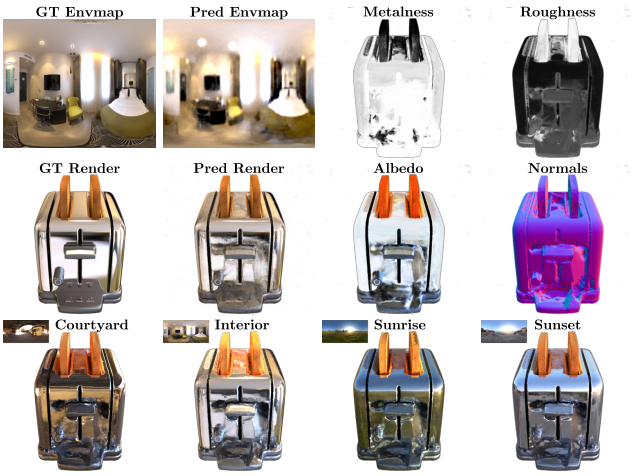

Figure 19. **Qualitative results on the Shiny Blender 'toaster' scene.**