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 physically 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 accurately model 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 attains state-of-the-art relighting quality after only $\sim$1 hour of training in a single NVIDIA A100 GPU.

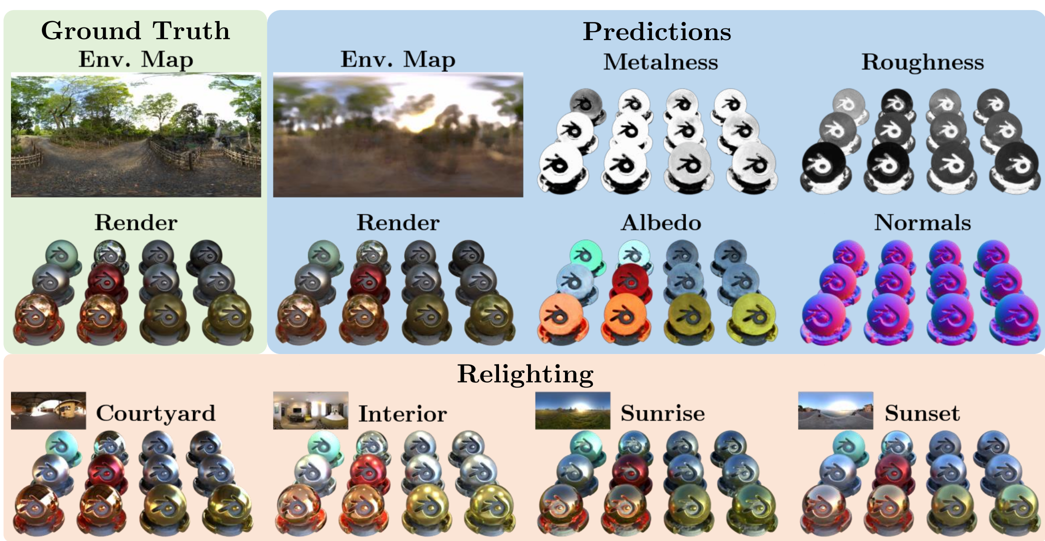

Figure 1: We visualize the lighting, material properties (albedo, metalness, and roughness), and geometry predicted by our model in addition to four relighting predictions of the 'materials' scene. Our method predicts high-frequency illumination with only $\sim$1 hour of training thus enabling the efficient digitization of relightable objects.

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

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

creating photo-realistic environments [6, 28, 21], as well as capturing real-world environments in a way that allows us to render new photo-realistic views [25, 32]. 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.

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) [18, 1, 34], 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 [29, 3, 16, 10, 41]. 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 build upon these methods 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. The 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. While MLPs representing pre-integrated illumination have been previously exploited [14, 13], previous works do not supervise the MLP's learning, leading to physically inaccurate illumination representations. We propose a novel regularizer based on Monte Carlo sampling to ensure accurate learning of illumination.

However, the split sum approximation on its own does not take into account self-occlusions. This hinders material property estimation since shadows tend to be incorrectly attributed to objects' albedo. Thus, we derive an occlusion factor to alleviate the effect of self-occlusions. This factor is then approximated via Monte Carlo sampling and used to supervise an occlusion MLP.

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 pre-integrated illumination as a single MLP with a corresponding regularization to ensure learning a physically-meaningful representation.

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

(iii) We demonstrate the effectiveness of our method in extracting environmental lighting, geometry, and material properties by achieving competitive reconstruction and relighting quality on both synthetic and real data with under one hour of training on a single NVIDIA A100 GPU.

2 Related works

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, 34, 19, 4, 26] 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, 31]. Signed Distance Function (SDF)-based methods [36, 40, 37, 12] 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 [36] but reformulates the radiance prediction to disentangle environment lighting and material properties.

2.2 Neural inverse rendering

The task of inverse rendering is a long-standing problem in computer graphics which consists of estimating the properties of a 3D scene such as shape, material, and lighting from a set of image observations. The success of neural rendering methods for novel view rendering and 3D reconstruction has led to a variety of works [2, 45, 43, 29, 3, 20, 46, 42, 14, 16, 30] exploiting neural rendering for inverse rendering. Due to the challenging nature of this problem, multiple simplifying assumptions have been adopted. Some works simplify the modeling of lighting by using low-frequency representations such as spherical gaussians [2, 45, 43, 29, 46, 39, 10] or low-resolution environment maps [2, 43, 46]. 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, 20, 17, 13, 14]. These works represent pre-integrated lighting with an autoencoder-based illumination network [3, 13], with a set of learnable images for different roughness levels [20, 17], or with an MLP with integrated spherical harmonic encoding as input [14]. Autoencoder-based methods rely on learnt illumination features incompatible with existing rendering pipelines. Learnable images are susceptible to noise and require re-integrating illumination whenever the base illumination is updated. Lastly, using integrated encodings to avoid integrating light leads to a representation that is not physically based. 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 physically–based pre-integrated lighting. An issue arising from the split sum approximation is that pre-integrated illumination is blind to geometry and does not account for the occlusion of light sources due to geometry 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 outgoing 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

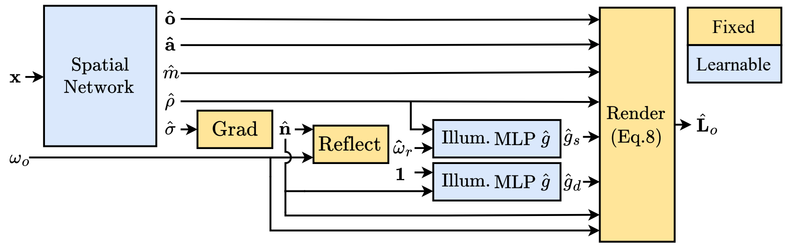

Figure 2: **Proposed architecture.** A spatial network maps spatial coordinates $\mathbf{x}$ into geometry (σ), material properties (albedo $\hat{\mathbf{a}}$, metalness $\hat{m}$, and roughness $\hat{\rho}$), and occlusion factors ($\hat{\mathbf{o}}$). The pre-integrated illumination MLP predicts both specular $\hat{g}_s(\hat{\omega}_r, \hat{\rho})$ and diffuse $\hat{g}_d(\hat{\mathbf{n}}, \rho = 1)$ terms by using the predicted normals $\hat{\mathbf{n}}$, roughness, and the reflection vector $\hat{\omega}_r$ of view direction ω_o. Finally, the specular and diffuse terms are combined with material properties to compute output radiance $\hat{\mathbf{L}}_o$.

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:

$$\hat{\mathbf{C}}(\mathbf{r};\theta) = \int_{t_n}^{t_f} T(t)\,\hat{\sigma}(\mathbf{r}(t))\,\hat{L}_o(\mathbf{r}(t),\omega_o)\,\mathrm{d}t, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \hat{\sigma}(\mathbf{r}(s))\,\mathrm{d}s\right). \quad (1)$$

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:

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

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 [36]. Learning a valid SDF requires the use of an additional Eikonal loss term $\mathcal{L}_{\text{Eik}}$. For more details, please refer to [36].

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 fig. 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:

$$\mathbf{L}_o = \int_{\Omega} (\mathbf{k}_d\frac{\mathbf{a}}{\pi} + \mathbf{f}_s)\mathbf{L}_i\langle\omega_i,\mathbf{n}\rangle d\omega_i, \quad (3)$$

where $\mathbf{L}_i$ is the incoming radiance, $\mathbf{a}$ is the object'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 notation the dependency of incoming radiance on ω_i as well as the dependency of material properties on position $\mathbf{x}$. Radiance $\mathbf{L}_o$ has diffuse and specular components $\mathbf{L}_d$ and $\mathbf{L}_s$. Image-based lighting methods often employ the split sum approximation to calculate specular lighting $\mathbf{L}_s$ by splitting the integral into two components: one containing the incoming light $\mathbf{L}_i$, and one depending only on material properties. We use the Disney [11] microfacet BRDF parameterized by albedo, metalness, and roughness. The specular component is modeled with the Cook-Torrance GGX [33, 35] BRDF, leading to the following approximation:

$$\mathbf{L}_s \approx \frac{\int_\Omega D(\omega_i, \omega_r, \rho)\mathbf{L}_i \langle \omega_i, \mathbf{n}\rangle d\omega_i}{\int_\Omega D(\omega_i, \omega_r, \rho)\langle \omega_i, \mathbf{n}\rangle d\omega_i} \int_\Omega \mathbf{f}_s \langle \omega_i, \mathbf{n}\rangle d\omega_i = g(\omega_r, \rho) \int_\Omega \mathbf{f}_s \langle \omega_i, \mathbf{n}\rangle d\omega_i, \tag{4}$$

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 as in [11] since it depends only on the BRDF and not a scene's lighting. The term on the left in Equation (4) depends on the scene's lighting and the microfacet distribution function $D(\omega_i, \omega_r, \rho)$. The following sections refer to this term as $g(\omega_r, \rho)$ and aim to estimate it with an MLP representation. As shown in fig. 2, we estimate an object's albedo $\hat{\mathbf{a}}$, metalness $\hat{m}$, and roughness $\hat{\rho}$ from the spatial network.

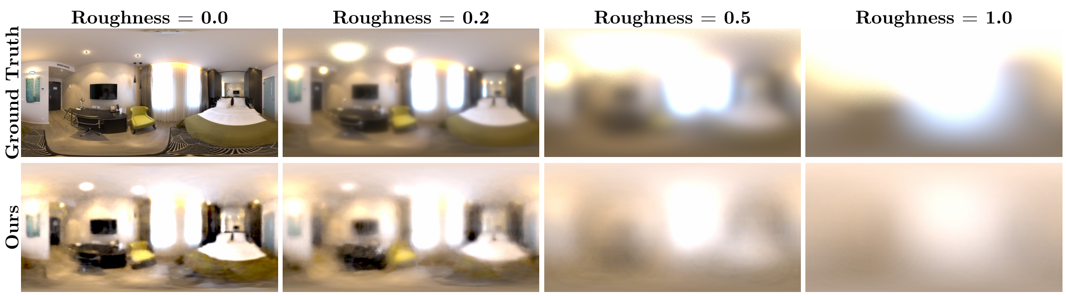

Figure 3: **Pre-integrated environment illumination.** We visualize the pre-integrated illumination $\hat{g}(\omega_r, \rho)$ for varying roughness values along our model's prediction for the 'toaster' scene. Our pre-integrated illumination MLP accurately approximates pre-integrated lighting across roughness values thanks to our novel regularization loss based on Monte Carlo sampling.

3.3 Pre-integrated illumination MLP representation

We propose to estimate the pre-integrated lighting $g(\omega_r, \rho)$ at different roughness levels through a pre-integrated illumination MLP $\hat{g}(\omega_r, \rho)$. Please refer to Appendix A.2 for details on how $\hat{g}(\omega_r, \rho)$ is used to calculate $\hat{\mathbf{L}}_d$ and $\hat{\mathbf{L}}_s$. 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 $\hat{\mathbf{L}}_i(\omega)$ which can be extracted from $\hat{g}$ querying perfect specular reflections $\hat{g}(\omega, 0)$.

$$\mathcal{L}_\mathrm{D}(\theta) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \|\hat{g}(s) - \bar{g}(s)\|_2^2, \text{ with } \bar{g}(s) = \frac{\sum_{\omega_i \in \Omega} D(\omega_i, \omega_s, \rho_s)\hat{g}(\omega_i, 0)\langle \omega_i, \omega_s\rangle}{\sum_{\omega_i \in \Omega} D(\omega_i, \omega_s, \rho_s)\langle \omega_i, \omega_s\rangle}, \tag{5}$$

where the set $\mathcal{S}$ consists of paired samples of directions ω_s and roughness ρ_s. Directions are taken uniformly on a sphere, and half the roughness samples are taken uniformly in the range $[0, 1]$ with the other half fixed to 1 to ensure correct learning of diffuse lighting. Please refer to Appendix A.1 for a detailed derivation of $\bar{g}(s)$. 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 fig. 3 for a specific scene.

3.4 Occlusion factors

The split sum approximation does not consider the occlusion of light sources due to geometry. Occlusions can be incorporated by multiplying incoming light $\mathbf{L}_i$ by a binary visibility function V_i:

$$\mathbf{L}_o^V = \int_\Omega (\mathbf{k}_d \frac{\mathbf{a}}{\pi} + \mathbf{f}_s)\mathbf{L}_i V_i \langle \omega_i, \mathbf{n}\rangle d\omega_i, \tag{6}$$

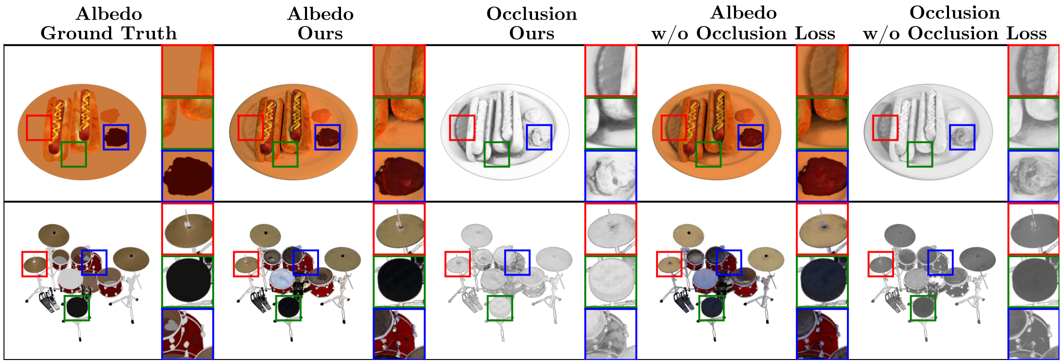

Figure 4: **Occlusion loss visualization.** We visualize the albedo and occlusion predicted by our method with and without the proposed occlusion regularization loss. When no regularization is used, we observe that the occlusion prediction fails at disentangling shadows from the albedo. Additionally, darker materials might wind up with lighter albedos due to occlusion overcompensation.

with V_i taking a value of 1 when there are no occlusions and 0 when incoming light is occluded by geometry. Both diffuse and specular integrals can be rewritten to incorporate visibility via occlusion factors $\mathbf{o}_d(\mathbf{x})$ and $\mathbf{o}_s(\mathbf{x})$ which multiply the split sum diffuse and specular lighting terms respectively. We propose learning the occlusion factors $\mathbf{o}(\mathbf{x})$ with an MLP by supervising them with Monte Carlo estimates $\bar{\mathbf{o}}(\mathbf{x})$ using the predicted geometry. Please refer to Appendix A.3 for the derivation of Monte Carlo estimates $\bar{\mathbf{o}}(\mathbf{x})$. Given the Monte Carlo estimates, we supervise the predicted occlusion terms $\hat{\mathbf{o}}(\mathbf{x})$ as follows:

$$\mathcal{L}_{\mathrm{o}}(\theta) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} w \left\| \hat{\mathbf{o}}(\mathbf{x}) - \bar{\mathbf{o}}(\mathbf{x}) \right\|_2^2 , \tag{7}$$

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 then calculated as follows:

$$\hat{\mathbf{L}}_o = \gamma(\hat{\mathbf{o}}_d * \hat{\mathbf{L}}_d + \hat{\mathbf{o}}_s * \hat{\mathbf{L}}_s), \tag{8}$$

where γ maps 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 (7). That is,

$$\mathcal{L}_{\mathrm{m}}(\theta) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} w \left\| \hat{m}(\mathbf{x}) \right|_2^2 . \tag{9}$$

4 Experiments

4.1 Baselines

We compare against Nerfactor [45], NVDiffRec [20], NVDiffRecMC [8], NeRO [14], NMF [16], and TensoIR [10]. 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.

Table 1: **NeRFactor metrics.** We evaluate the reconstruction quality of our method against the baselines using 20 test images and 8 low-frequency illumination maps for each scene from the NeRFactor dataset. We scale albedo and relit images with a per-channel factor before computing metrics. Our method attains competitive performance across all metrics with a low runtime.

Method	Normals	Albedo			Relighting			Average
	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Runtime
NerFactor	30.49	23.53	0.910	0.109	23.66	0.895	0.120	>20 hr.
NVDiffRec	26.47	23.05	0.901	0.123	21.88	0.880	0.111	0.98 hr.
NVDiffRecMC	25.98	23.84	0.918	0.114	24.06	0.902	0.099	2.95 hr.
NeRO	30.59	22.83	0.897	0.117	23.68	0.907	0.093	18.38 hr.
NMF	24.14	-	-	-	22.23	0.895	0.093	2.91 hr.
TensoIR	22.90	25.21	**0.929**	**0.087**	23.78	0.907	0.100	3.53 hr.
Ours	**17.52**	**25.29**	0.924	0.108	**27.31**	**0.941**	**0.061**	**0.81 hr.**

Table 2: **Blender and Shiny Blender metrics.** We report the average of relighting reconstruction metrics and normal error for our extended Blender and Shiny Blender datasets. Metrics are computed as the average of 20 test views across 7 high-frequency illumination conditions for each scene. We scale images by a per-channel factor for relighting metrics. Our method outperforms the baselines across all metrics for the Blender dataset and has a higher PSNR for the Shiny Blender dataset.

Method	Blender				Shiny Blender			
	Normals	Relighting			Normals	Relighting		
	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NVDiffRec	26.52	20.11	0.857	0.138	23.64	21.39	0.848	0.177
NVDiffRecMC	24.74	22.50	0.884	0.136	13.75	24.60	**0.911**	0.151
NeRO	31.59	18.47	0.847	0.158	**04.11**	16.82	0.844	0.203
NMF	20.66	21.21	0.881	0.118	06.85	24.20	0.908	**0.136**
TensoIR	18.05	22.58	0.891	0.120	13.14	22.33	0.840	0.193
Ours	**16.18**	**22.73**	**0.906**	**0.106**	09.07	**24.96**	0.904	0.144

4.2 Experimental setup

Datasets. We report results using the NeRFactor [45] dataset along with extended versions of the NeRF Blender [18] (Blender) and the RefNeRF Shiny Blender [34] (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. Finally, we report qualitative results on real-world objects using the CO3D [24] dataset. Each object in the dataset consists of a set of images captured along a circular path along with automatically extracted foreground segmentation masks. We estimate each image's camera pose with Colmap [27].

Relighting evaluation. 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. We utilize the same Blender rendering pipeline to compute relighting metrics for baselines where explicit meshes and material properties are extracted. 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 Index Measure (SSIM) [38], and Learned Perceptual Image Patch Similarity (LPIPS) [44] 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. Our method attains competitive relighting performance while maintaining a low runtime.

Figure 5: **Qualitative real-world results.** We present qualitative results on four scenes from the CO3D dataset. Our method can successfully recover object geometry, material properties, and illumination even for challenging scenes captured in the wild.

Albedo evaluation. In addition to overall relighting quality, we evaluate the ability of our method to recover albedo. We report reconstruction metrics (PSNR, SSIM, and LPIPS) on the predicted albedo in table 1. 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 do not report albedo metrics for other datasets due to the lack of ground truth. We exclude results from NMF [16] since the albedo in their lighting formulation is not comparable to the other methods. Thanks to our proposed occlusion factor and material regularization, our method is on average better able to reconstruct albedo.

Normals evaluation. We measure the pixel-wise Mean Absolute Error (MAE) between ground truth and predicted normal images to evaluate geometric reconstruction quality. The MAE is weighted by ground truth alpha values to lower the effect of prediction errors at object borders. Our method recovers a good estimate of geometry for most scenes as evidenced in tables 1 and 2.

Real-world qualitative results. In real-world scenes, the far-field illumination assumption is violated and objects don't follow any specific BRDF model as opposed to synthetic scenes. Both of these differences make inverse rendering from real-world data a much more challenging task than with synthetic data. Therefore, we provide qualitative results in fig. 5 to validate our method on real-world captures from the CO3D dataset. It can be observed that even in this challenging scenario our method is capable of recovering accurate object geometry, as well as providing a reasonable estimation of material properties and environment maps.

5 Discussions

5.1 Ablations

Occlusion loss. We visualize the effects of the proposed occlusion loss in fig. 4. 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 hotdog example. Quantitatively, we measure the importance of adding the occlusion loss to our model in table 3, where it improves both relighting and albedo reconstruction.

Occlusion averaging. The occlusion factor we derive is a per-channel factor that depends on estimated lighting. However, since both are being learned jointly, we observe that training can be noisy. We find in table 3 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 only white light with varying intensities, which reduces noise during training and uses fewer parameters.

Table 3: **NeRFactor ablation results.** We report reconstruction and relighting metrics for different variations of our methodology on the NeRFactor dataset. While the proposed regularizations do not have a noticeable effect on geometry, they all lead to improvements in albedo and relighting quality.

Method	Normals	Albedo			Relighting		
	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Ours (w/o Occ. Avg.)	**17.23**	24.73	0.919	0.109	27.25	0.941	0.061
Ours (w/o Occ. Loss)	17.29	22.13	0.900	0.122	26.40	0.936	0.064
Ours (w/o Met. Reg.)	17.82	23.92	0.916	0.123	26.56	0.936	0.066
Ours	17.52	**25.29**	**0.924**	**0.108**	**27.31**	**0.941**	**0.061**

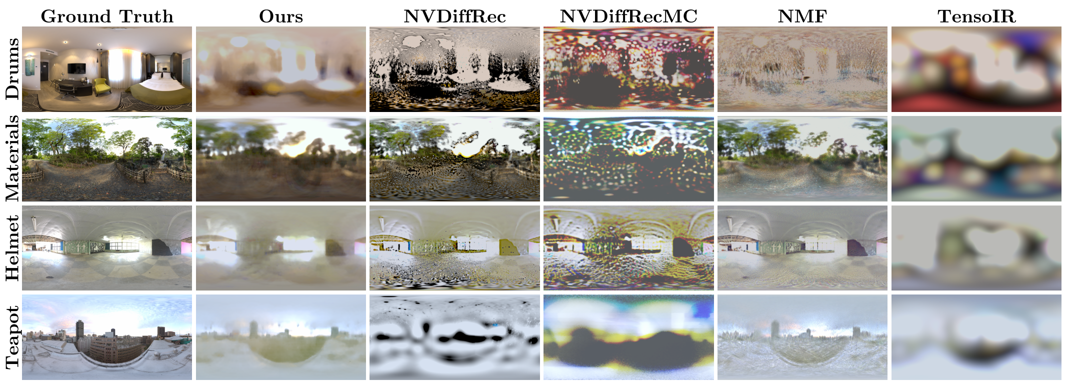

Figure 6: **Blender and Shiny Blender illumination visualizations.** We visualize the predicted illumination environment maps for our method and baselines for two scenes in the Blender dataset and two scenes in the Shiny Blender dataset. Illumination is scaled by a per-channel factor to account for albedo-illumination ambiguity. Our proposed illumination inherits smoothness from the MLP representation but still captures high-quality details such as trees.

Material regularization. We measure the effect of the material regularization in table 3. Penalizing metalness prediction discourages our model from explaining radiance through environment lighting with overpredicted metallic surfaces. This leads to improved albedo predictions as shown in table 3. However, as visualized in fig. 1, the loss coefficient is small enough such that our model is still capable of correctly predicting metallic surfaces.

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. Physical accuracy of our pre-integrated MLP representation is ensured thanks to the proposed illumination regularization. Additionally, we define occlusion factors for diffuse and specular lighting so that self-occlusions are accounted for with the split sum approximation. Finall, we propose a way of supervising occlusion MLPs to learn the proposed occlusion estimators. Altogether, our method produces 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, our method comes with some limitations. The major assumptions we rely on come from using image-based lighting, the split sum approximation, and Monte Carlo sampling. 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 disregard the effect of indirect illumination. This has a noticeable impact on albedo for reflective surfaces such as the 'toaster' scene. Additionally, we only consider the reflection of light and don't model transmission and subsurface scattering effects. Finally, we use a low number of uniform Monte Carlo samples for the occlusion loss leading to errors due to strong and small light sources. This is mostly noticeable in albedo predictions for objects in the synthetic Blender dataset, where shadows can still be noticed in the albedo predictions. We hope future works will tackle these limitations.

7 Acknowledgements

The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST) - Center of Excellence for Generative AI, under award number 5940.

References

[1] Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. ICCV (2021)

[2] Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.P.: Nerd: Neural reflectance decomposition from image collections. In: IEEE International Conference on Computer Vision (ICCV) (2021)

[3] Boss, M., Jampani, V., Braun, R., Liu, C., Barron, J.T., Lensch, H.P.: Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)

[4] Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)

[5] Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), `http://www.blender.org`

[6] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)

[7] Guo, Y.C.: Instant neural surface reconstruction (2022), https://github.com/bennyguo/instant-nsr-pl

[8] Hasselgren, J., Hofmann, N., Munkberg, J.: Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. arXiv:2206.03380 (2022)

[9] Jiang, Y., Yin, S., Li, K., Luo, H., Kaynak, O.: Industrial applications of digital twins. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **379**, 20200360 (08 2021). https://doi.org/10.1098/rsta.2020.0360

[10] Jin, H., Liu, I., Xu, P., Zhang, X., Han, S., Bi, S., Zhou, X., Xu, Z., Su, H.: Tensoir: Tensorial inverse rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023)

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

[12] Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023)

[13] Liang, R., Chen, H., Li, C., Chen, F., Panneer, S., Vijaykumar, N.: Envidr: Implicit differentiable renderer with neural environment lighting. arXiv preprint arXiv:2303.13022 (2023)

[14] Liu, Y., Wang, P., Lin, C., Long, X., Wang, J., Liu, L., Komura, T., Wang, W.: Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. In: SIGGRAPH (2023)

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

[16] Mai, A., Verbin, D., Kuester, F., Fridovich-Keil, S.: Neural microfacet fields for inverse rendering (2023)

[17] Mao, S., Wu, C., Shen, Z., Zhang, L.: Neus-pir: Learning relightable neural surface using pre-integrated rendering. CoRR **abs/2306.07632** (2023). https://doi.org/10.48550/ARXIV.2306.07632, `https://doi.org/10.48550/arXiv.2306.07632`

[18] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)

[19] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)

[20] Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting Triangular 3D Models, Materials, and Lighting From Images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8280–8290 (June 2022)

[21] Müller, M., Casser, V., Lahoud, J., Smith, N., Ghanem, B.: Sim4cv: A photo-realistic simulator for computer vision applications. International Journal of Computer Vision **126**(9), 902–919 (2018). https://doi.org/10.1007/s11263-018-1073-7

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

[23] Pharr, M., Jakob, W., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edn. (2016)

[24] Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., Novotny, D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In: International Conference on Computer Vision (2021)

[25] Rematas, K., Liu, A., Srinivasan, P.P., Barron, J.T., Tagliasacchi, A., Funkhouser, T., Ferrari, V.: Urban radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12932–12942 (2022)

[26] Rojas, S., Zarzar, J., Pérez, J.C., Sanakoyeu, A., Thabet, A., Pumarola, A., Ghanem, B.: Re-rend: Real-time rendering of nerfs across devices. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3632–3641 (October 2023)

[27] Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4104–4113 (2016)

[28] Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and Service Robotics (2017), `https://arxiv.org/abs/1705.05065`

[29] Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: CVPR (2021)

[30] Sun, C., Cai, G., Li, Z., Yan, K., Zhang, C., Marshall, C., Huang, J.B., Zhao, S., Dong, Z.: Neural-pbir reconstruction of shape, material, and illumination. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 18046–18056 (October 2023)

[31] Sun, J., Chen, X., Wang, Q., Li, Z., Averbuch-Elor, H., Zhou, X., Snavely, N.: Neural 3d reconstruction in the wild. In: ACM SIGGRAPH 2022 Conference Proceedings. pp. 1–9 (2022)

[32] Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P., Barron, J.T., Kretzschmar, H.: Block-NeRF: Scalable large scene neural view synthesis. arXiv (2022)

[33] Torrance, K.E., Sparrow, E.M.: Theory for off-specular reflection from roughened surfaces∗. J. Opt. Soc. Am. **57**(9), 1105–1114 (Sep 1967). https://doi.org/10.1364/JOSA.57.001105, `https://opg.optica.org/abstract.cfm?URI=josa-57-9-1105`

[34] Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. CVPR (2022)

[35] Walter, B., Marschner, S.R., Li, H., Torrance, K.E.: Microfacet models for refraction through rough surfaces. In: Proceedings of the 18th Eurographics Conference on Rendering Techniques. p. 195–206. EGSR'07, Eurographics Association, Goslar, DEU (2007)

[36] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)

[37] Wang, Y., Skorokhodov, I., Wonka, P.: Hf-neus: Improved surface reconstruction using high-frequency details. Advances in Neural Information Processing Systems **35**, 1966–1978 (2022)

[38] Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004). https://doi.org/10.1109/TIP.2003.819861

[39] Wu, H., Hu, Z., Li, L., Zhang, Y., Fan, C., Yu, X.: Nefii: Inverse rendering for reflectance decomposition with near-field indirect illumination. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4295–4304 (2023). https://doi.org/10.1109/CVPR52729.2023.00418

[40] Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: Thirty-Fifth Conference on Neural Information Processing Systems (2021)

[41] Zhang, J., Yao, Y., Li, S., Liu, J., Fang, T., McKinnon, D., Tsin, Y., Quan, L.: Neilf++: Inter-reflectable light fields for geometry and material estimation. International Conference on Computer Vision (ICCV) (2023)

[42] Zhang, K., Luan, F., Li, Z., Snavely, N.: 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. pp. 5565–5574 (2022)

[43] Zhang, K., Luan, F., Wang, Q., Bala, K., Snavely, N.: 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)

[44] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 586–595. IEEE Computer Society, Los Alamitos, CA, USA (jun 2018). https://doi.org/10.1109/CVPR.2018.00068, `https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00068`

[45] Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. ACM Transactions on Graphics (TOG) (2021)

[46] Zhang, Y., Sun, J., He, X., Fu, H., Jia, R., Zhou, X.: Modeling indirect illumination for inverse rendering. In: CVPR (2022)

A Appendix

A.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 eq. (5). We first split the specular light integral into two terms:

$$\mathbf{L}_s = \frac{\int\limits_\Omega \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int\limits_\Omega \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i} \int\limits_\Omega \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i. \tag{10}$$

As mentioned in section 3.2, the term on the right can be precomputed so we focus on calculating an approximation for the term on the left, which we denote $g(\omega_r, \rho)$.

$$g(\omega_r, \rho) = \frac{\int\limits_\Omega \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int\limits_\Omega \mathbf{f}_s \langle \omega_i, \mathbf{n} \rangle d\omega_i}. \tag{11}$$

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

$$\mathbf{f}_s = \frac{DFG}{4\langle \omega_\mathbf{o}, \mathbf{n} \rangle \langle \omega_\mathbf{i}, \mathbf{n} \rangle}, \tag{12}$$

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

$$\mathbf{f}_s \approx \frac{D}{4\langle\omega_o, \mathbf{n}\rangle}, \quad g(\omega_r, \rho) \approx \frac{\int\limits_\Omega D\mathbf{L}_i\langle\omega_i, \mathbf{n}\rangle d\omega_i}{\int\limits_\Omega D\langle\omega_i, \mathbf{n}\rangle d\omega_i}, \tag{13}$$

as shown in Equation (4). The GGX (Trowbridge-Reitz) microfacet distribution function D is defined as:

$$D(\omega_i, \omega_o, \mathbf{n}, \rho) = \frac{\rho^2}{\pi(\langle\mathbf{h}, \mathbf{n}\rangle^2(\rho^2 - 1) + 1)^2}, \tag{14}$$

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:

$$D(\omega_i, \omega_r, \rho) \approx \frac{\rho^2}{\pi(\frac{1+\langle\omega_i, \omega_r\rangle}{2}(\rho^2 - 1) + 1)^2}, \tag{15}$$

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 following expression:

$$g(\omega_r, \rho) \approx \frac{\sum\limits_\Omega D(\omega_i, \omega_r, \rho)\mathbf{L}_i\langle\omega_i, \omega_r\rangle}{\sum\limits_\Omega D(\omega_i, \omega_r, \rho)\langle\omega_i, \omega_r\rangle}. \tag{16}$$

We finally obtain the expression for $\bar{g}$ in eq. (5) by replacing $\mathbf{L}_i$ with the estimates from $\hat{g}$ querying perfect specular reflections $\hat{g}(\omega, 0)$. A side effect of the approximations used is that for $\rho = 1$ in eq. (15), we have that $D(\omega_i, \omega_r, 1) \approx \frac{1}{\pi}$ and together with eq. (13) we obtain that

$$g(\mathbf{n}, 1) \approx \frac{1}{\pi}\int\limits_\Omega \mathbf{L}_i\langle\omega_i, \mathbf{n}\rangle d\omega_i. \tag{17}$$

This allows us to reuse the same network $\hat{g}$ used to approximate $\mathbf{L}_s$ also approximate $\mathbf{L}_d$ as follows:

$$\hat{\mathbf{L}}_d = \hat{g}(\hat{\mathbf{n}}, 1)\hat{\mathbf{k}}_d\hat{\mathbf{a}}. \tag{18}$$

A.2 BRDF details

The only remaining terms to compute for obtaining diffuse and specular components are the precomputed BRDF integral and the diffuse coefficient $\mathbf{k}_d$ which we compute following [11]. Given our material property estimates we can compute $\mathbf{k}_d$ as follows:

$$\begin{aligned} \hat{\mathbf{F}}_0 &= (1 - \hat{m}) * 0.04 + \hat{m} * \hat{\mathbf{a}}, \\ \hat{\mathbf{F}}_r &= \hat{\mathbf{F}}_0 + (1 - \hat{\rho} - \hat{\mathbf{F}}_0) * (1 - \langle\hat{\mathbf{n}}, \omega_o\rangle)^5, \\ \hat{\mathbf{k}}_d &= (1 - \hat{m}) * (1 - \hat{\mathbf{F}}_r). \end{aligned} \tag{19}$$

Finally, we store two coefficients F_1 and F_2 in a two-dimensional lookup table as per [11] and compute the BRDF integral as follows:

$$\int\limits_\Omega \mathbf{f}_s\langle\omega_i, \mathbf{n}\rangle d\omega_i = \hat{\mathbf{F}}_r * F_1 + F_2. \tag{20}$$

The final expressions for the diffuse and specular coefficients are thus

$$\hat{\mathbf{L}}_d = \hat{g}(\hat{\mathbf{n}}, 1)\hat{\mathbf{k}}_d\hat{\mathbf{a}}, \quad \hat{\mathbf{L}}_s = \hat{g}(\hat{\omega}_r, \hat{\rho}) * (\hat{\mathbf{F}}_r * F_1 + F_2). \tag{21}$$

A.3 Derivation of occlusion factor approximation

We now go over the derivation of the occlusion factor Monte Carlo approximation. We aim to approximate occlusion factors $\mathbf{o}_d(\mathbf{x})$ and $\mathbf{o}_s(\mathbf{x})$ such that the diffuse/specular components with visibility, $\mathbf{L}_d^V$ and $\mathbf{L}_s^V$, can be computed as a multiplication between the diffuse/specular components without visibility, $\mathbf{L}_d$ and $\mathbf{L}_s$, and their respective occlusion factors. For the diffuse component,

$$\mathbf{L}_d^V = \mathbf{k}_d \frac{\mathbf{a}}{\pi} \int_\Omega \mathbf{L}_i V_i \langle \omega_\mathbf{i}, \mathbf{n} \rangle d\omega_i = \frac{\int_\Omega \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int_\Omega \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i} \mathbf{k}_d \frac{\mathbf{a}}{\pi} \int_\Omega \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i. \tag{22}$$

$$\text{Thus, } \mathbf{L}_d^V = \mathbf{o}_d(\mathbf{x}) \mathbf{L}_d \text{ with } \mathbf{o}_d(\mathbf{x}) = \frac{\int_\Omega \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int_\Omega \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}. \tag{23}$$

To incorporate this information, we learn this factor for the diffuse radiance. We propose learning the occlusion factor $\mathbf{o}_d(\mathbf{x})$ with an MLP, supervising it with Monte Carlo estimates $\bar{\mathbf{o}}_d(\mathbf{x})$ using the predicted geometry. We approximate $\mathbf{o}_d(\mathbf{x})$ with Monte Carlo sampling by taking the same number of ω_i samples for both integrals:

$$\bar{\mathbf{o}}_d(\mathbf{x}) = \frac{\sum_{\omega_i \in \Omega} \mathbf{L}_i V_i}{\sum_{\omega_i \in \Omega} \mathbf{L}_i}, \tag{24}$$

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:

$$\text{pdf}(\omega_i; \mathbf{n}) = \frac{\langle \omega_i, \mathbf{n} \rangle}{\pi}, \tag{25}$$

This cos-weighted sampling aids in reducing variance by eliminating the dot product factor from the estimation.

Similarly, for the specular component we have that

$$\mathbf{L}_s^V = \int_\Omega \mathbf{f}_s \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle d\omega_i = \frac{\int_\Omega \mathbf{f}_s \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int_\Omega \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i} \int_\Omega \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i. \tag{26}$$

$$\text{Thus, } \mathbf{L}_s^V = \mathbf{o}_s(\mathbf{x}) \mathbf{L}_s \text{ with } \mathbf{o}_s(\mathbf{x}) = \frac{\int_\Omega \mathbf{f}_s \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}{\int_\Omega \mathbf{f}_s \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle d\omega_i}. \tag{27}$$

We then use the approximation for $\mathbf{f}_s$ in eq. (13), leading to the following Monte Carlo estimate for $\bar{\mathbf{o}}_s(\mathbf{x})$:

$$\bar{\mathbf{o}}_s(\mathbf{x}) = \frac{\sum_{\omega_i \in \Omega} \mathbf{L}_i V_i \langle \omega_i, \mathbf{n} \rangle}{\sum_{\omega_i \in \Omega} \mathbf{L}_i \langle \omega_i, \mathbf{n} \rangle}, \tag{28}$$

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

$$\text{pdf}(\omega_i; \omega_r, \rho) = \frac{D(\omega_i, \omega_r, \rho) \langle \mathbf{n}, \omega_h \rangle}{4 \langle \mathbf{n}, \omega_o \rangle}, \tag{29}$$

which relies on the second approximation used in the previous section, and where ω_h is the half-vector angle between ω_i and ω_o.

A.4 Implementation details

We embed our proposed lighting decomposition within an efficient implementation of NeuS [7]. 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 with 60GB RAM and 6 CPU workers using an AMD EPYC 7713 64-Core processor for a total training time of $\sim$1 hour.

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 ~ 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 $(0, \inf)$.

A.5 Per-scene quantitative results

We report per-scene metrics for geometry, albedo, and relighting reconstruction using the NeRFactor dataset in tables 4 to 6. We report per-scene metrics for geometry and relighting using the Blender dataset in tables 7 to 10, and using the Shiny Blender dataset in tables 11 to 14. 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 figs. 7 to 14 and for the Shiny Blender dataset in figs. 15 to 19.

A.6 CO3D qualitative relighting results

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 CO3D dataset in figs. 20 to 23. Even in this challenging real-world setting where images are taken from a user-captured video, our method provides good-quality results.

A.7 Additional qualitative videos

Please refer to the included video files for additional qualitative videos showing predicted renders, material properties, and relighting for the Blender, Shiny Blender, and CO3D datasets. All videos follow the same camera trajectory at a fixed distance from the center of the scene. Please note that due to objects in CO3D lacking training images viewing the upper or lower surfaces of objects, the reconstruction quality at these locations suffers. This issue could be alleviated with better video-capturing trajectories.

Method	MAE ↓				
	avg.	**drums**	**ficus**	**hotdog**	**lego**
NerFactor	30.49	30.27	45.37	16.95	29.37
NVDiffRec	26.47	26.37	29.39	13.28	36.86
NVDiffRecMC	25.98	28.81	30.88	13.12	31.11
NeRO	30.59	22.00	50.69	19.78	29.90
NMF	24.14	20.62	39.00	10.85	26.10
TensoIR	22.90	**18.30**	36.38	14.21	**22.74**
Ours	**17.52**	18.33	**17.19**	**10.41**	24.13

Table 4: **NeRFactor per-scene MAE.**

Method	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
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
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
TensoIR	23.78	22.49	23.07	25.58	23.97	0.907	0.915	0.933	0.895	0.886	0.100	0.077	0.081	0.129	0.113
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 5: **NeRFactor per-scene relighting metrics.**

Method	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	23.05	19.47	23.67	28.20	20.87	0.901	0.880	0.938	0.942	0.844	0.123	0.118	0.090	0.110	0.174
NVDiffRecMC	23.84	20.28	22.16	**29.27**	23.65	0.918	0.892	0.923	**0.950**	0.905	0.114	0.118	0.105	0.100	0.134
NeRO	22.83	20.52	20.05	26.70	24.03	0.897	0.889	0.910	0.923	0.868	0.117	0.116	0.106	0.100	0.147
NMF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TensoIR	25.21	**24.48**	22.75	27.58	**26.05**	**0.929**	**0.936**	0.920	0.936	**0.925**	**0.087**	**0.058**	0.079	**0.088**	0.122
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 6: **NeRFactor per-scene albedo metrics.**

Method	MAE ↓							
	avg.	**chair**	**drums**	**ficus**	**hotdog**	**lego**	**materials**	**mic**	**ship**
NVDiffRec	26.52	20.71	28.70	28.96	15.46	40.53	22.37	21.39	34.04
NVDiffRecMC	24.74	16.47	28.69	38.18	16.36	34.84	12.04	21.04	30.27
NeRO	31.59	13.47	29.45	51.33	**13.13**	29.20	71.22	19.00	25.95
NMF	20.66	13.18	20.24	36.61	13.97	26.38	**08.69**	20.68	25.56
TensoIR	18.05	11.69	**17.88**	30.63	15.16	**19.18**	13.51	17.31	**19.02**
Ours	**16.18**	**11.65**	20.96	**17.38**	13.20	21.17	08.93	**14.37**	21.76

Table 7: **Blender per-scene MAE.**

Method	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
NeRO	18.47	22.73	13.51	21.12	19.99	21.81	12.73	17.82	18.04
NMF	21.21	22.53	21.38	22.62	20.52	22.05	24.87	18.34	17.41
TensoIR	22.58	**25.21**	22.10	23.90	22.01	**26.18**	23.00	18.79	**19.45**
Ours	**22.73**	25.00	**22.62**	**26.40**	20.94	23.88	**25.38**	18.75	18.88

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

Method	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
NeRO	0.847	0.905	0.774	0.912	0.899	0.856	0.755	0.920	0.754
NMF	0.881	0.908	0.890	0.934	0.890	0.863	0.913	0.926	0.722
TensoIR	0.891	0.929	0.899	0.935	0.891	**0.906**	0.871	0.934	0.763
Ours	**0.906**	**0.937**	**0.908**	**0.952**	0.914	0.901	**0.930**	**0.937**	**0.769**

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

Method	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
NeRO	0.158	0.099	0.229	0.088	**0.115**	0.132	0.218	0.095	0.291
NMF	0.118	0.093	0.103	0.066	0.135	0.108	0.081	0.087	0.271
TensoIR	0.120	0.082	0.102	0.076	0.148	**0.087**	0.130	0.088	**0.251**
Ours	**0.106**	**0.065**	**0.096**	**0.050**	0.116	0.097	**0.075**	**0.077**	0.269

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

Method	MAE ↓					
	avg.	**car**	**coffee**	**helmet**	**teapot**	**toaster**
NVDiffRec	23.64	40.01	21.46	18.14	07.73	30.86
NVDiffRecMC	13.75	10.83	23.34	12.73	08.77	13.08
NeRO	**04.11**	**05.72**	**04.72**	**01.29**	03.27	**05.54**
NMF	06.85	07.38	12.22	02.42	04.67	07.58
TensoIR	13.14	11.44	09.56	18.00	10.97	15.72
Ours	09.07	07.21	23.44	02.56	**02.29**	09.86

Table 11: **Shiny Blender per-scene MAE.**

Method	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
NeRO	16.82	15.36	17.21	14.09	25.31	12.12
NMF	24.20	24.58	17.88	**27.48**	30.66	**20.41**
TensoIR	22.33	25.31	18.42	19.45	31.35	17.14
Ours	**24.96**	**26.86**	18.70	21.51	**38.13**	19.62

Table 12: **Shiny Blender per-scene PSNR.**

Method	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
NeRO	0.844	0.837	0.867	0.835	0.964	0.717
NMF	0.908	0.917	0.843	**0.946**	0.982	**0.849**
TensoIR	0.840	0.899	0.857	0.784	0.970	0.692
Ours	0.904	**0.942**	0.883	0.877	**0.993**	0.827

Table 13: **Shiny Blender per-scene SSIM.**

Method	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
NeRO	0.203	0.152	0.248	0.259	0.049	0.305
NMF	**0.136**	0.092	0.208	**0.142**	0.032	**0.206**
TensoIR	0.193	0.125	0.203	0.277	0.048	0.313
Ours	0.144	**0.072**	0.204	0.195	**0.017**	0.234

Table 14: **Shiny Blender per-scene LPIPS.**

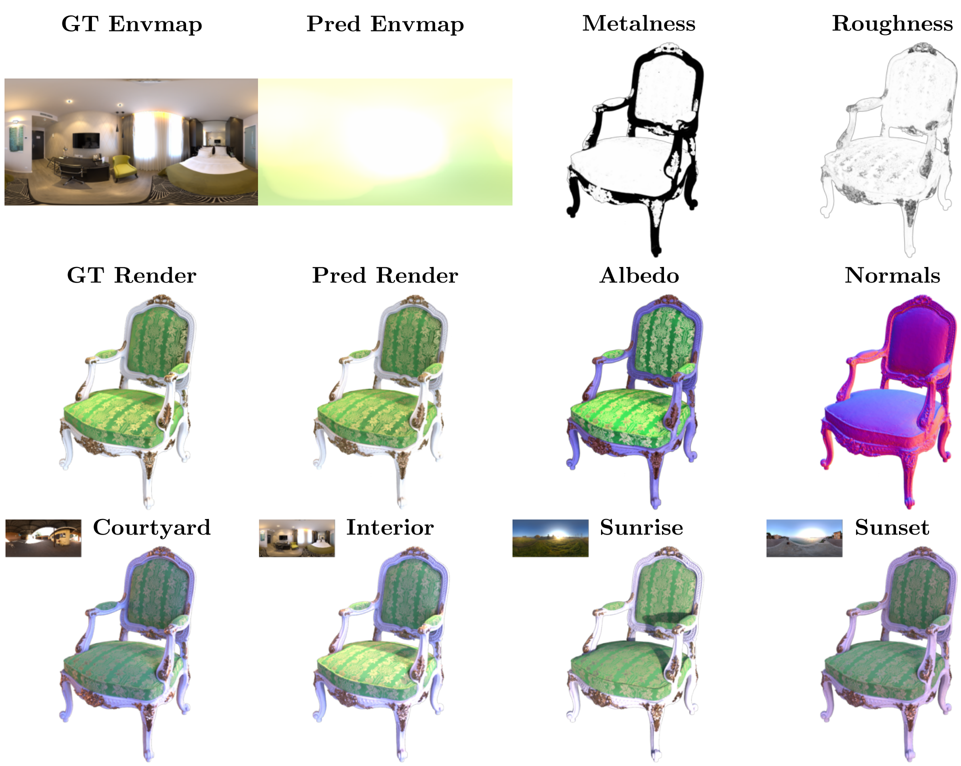

Figure 7: **Qualitative results on the Blender 'chair' scene.**

Figure 8: **Qualitative results on the Blender ‘drums’ scene.**

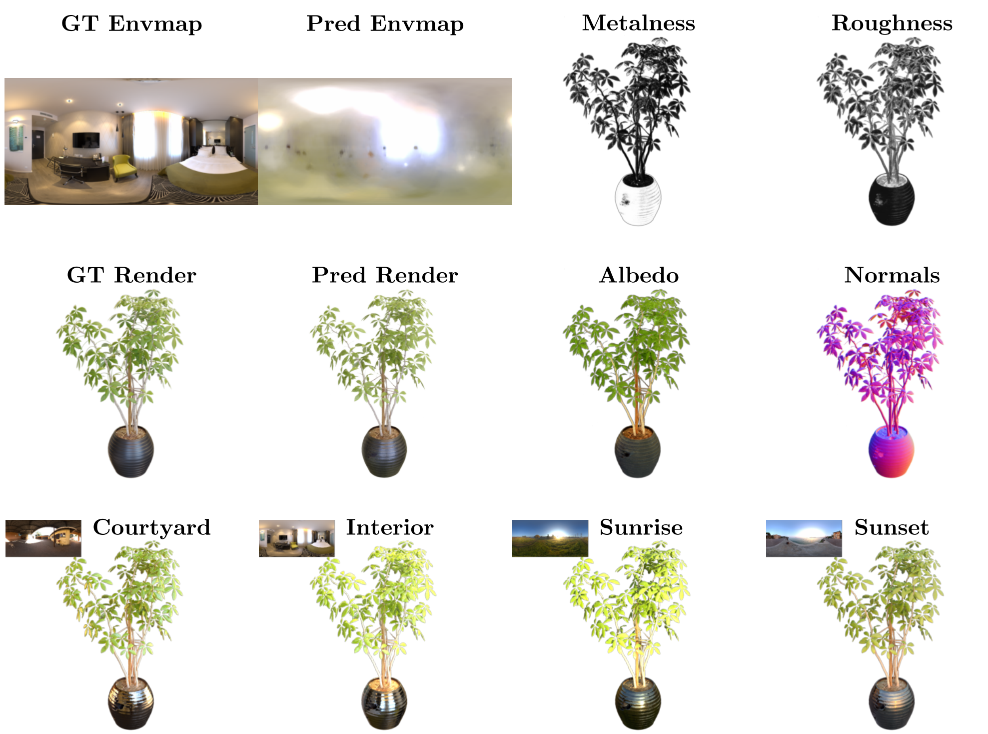

Figure 9: **Qualitative results on the Blender ‘ficus’ scene.**

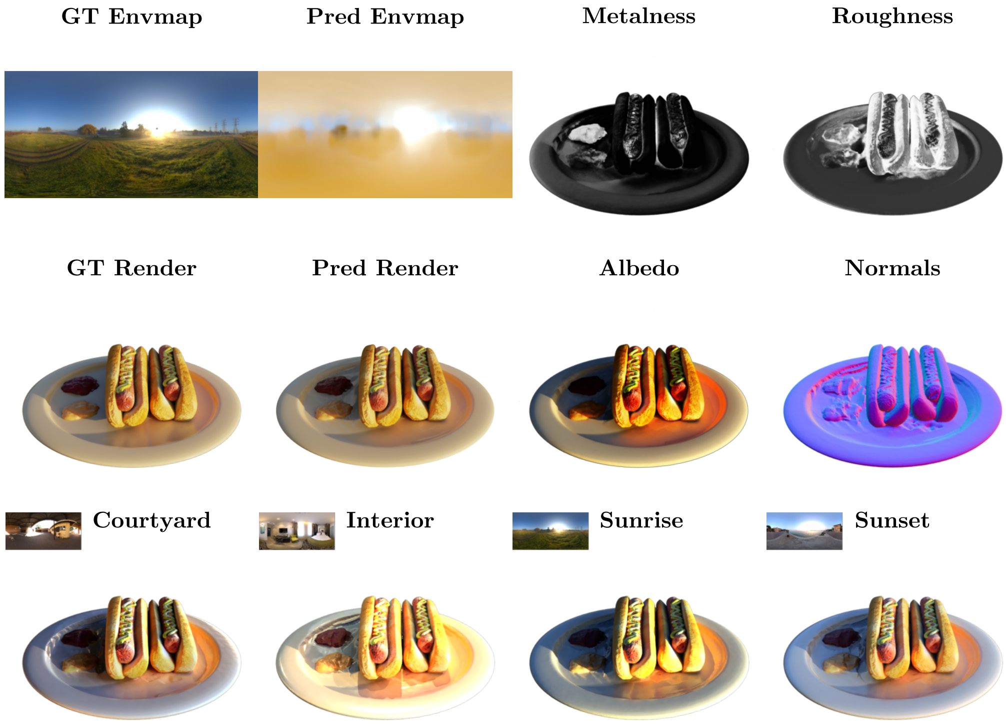

Figure 10: **Qualitative results on the Blender ‘hotdog’ scene.**

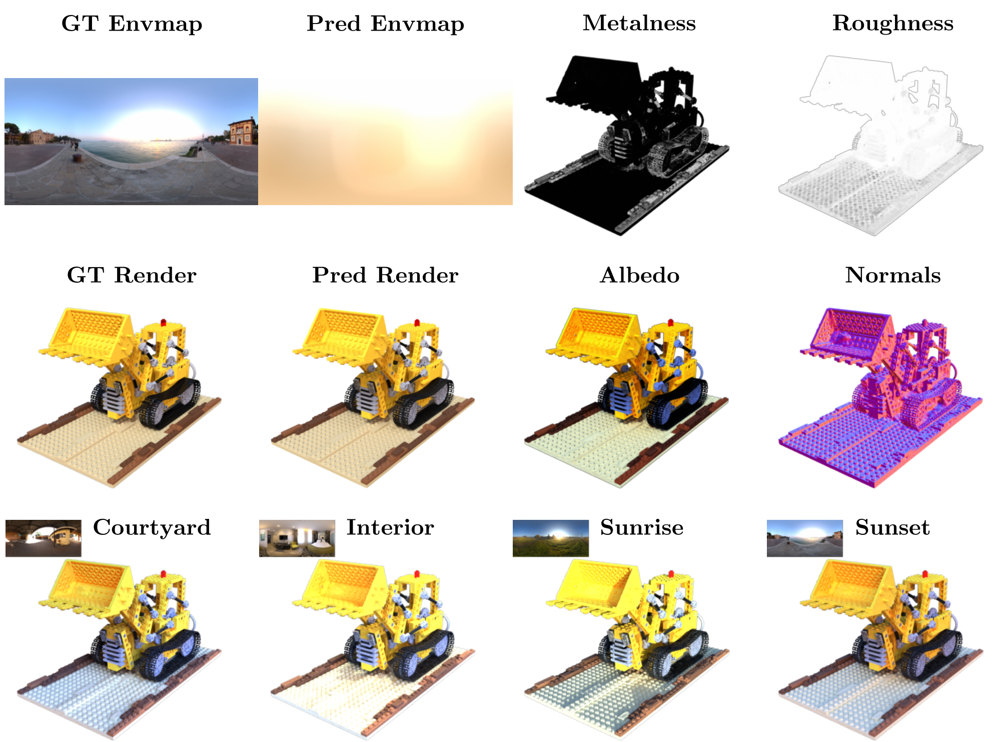

Figure 11: **Qualitative results on the Blender ‘lego’ scene.**

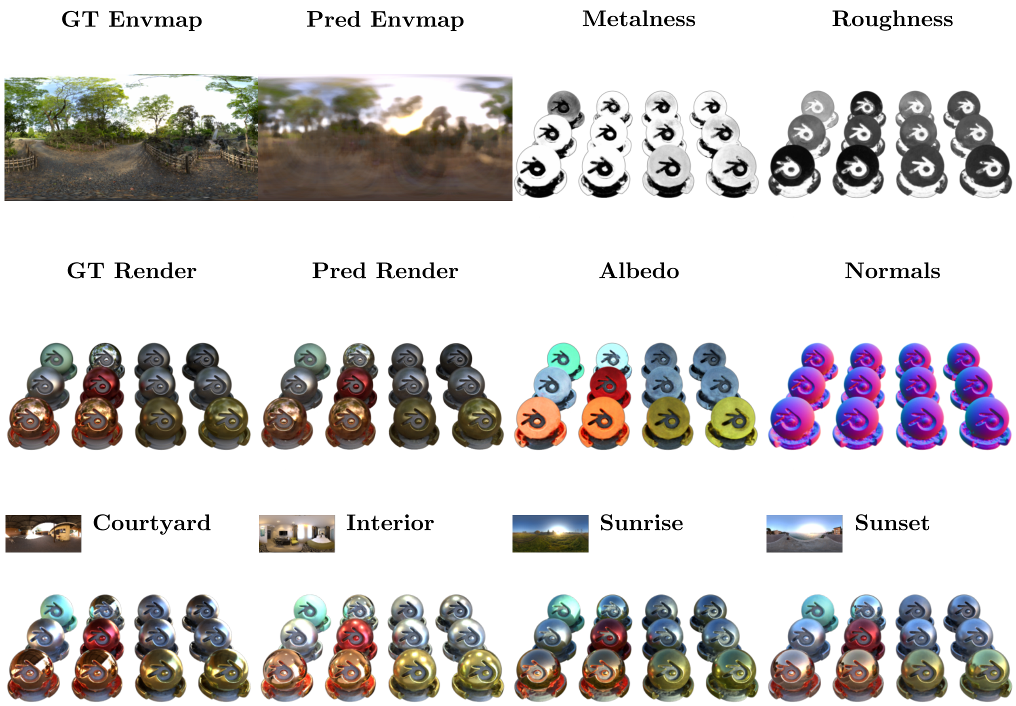

Figure 12: **Qualitative results on the Blender 'materials' scene.**

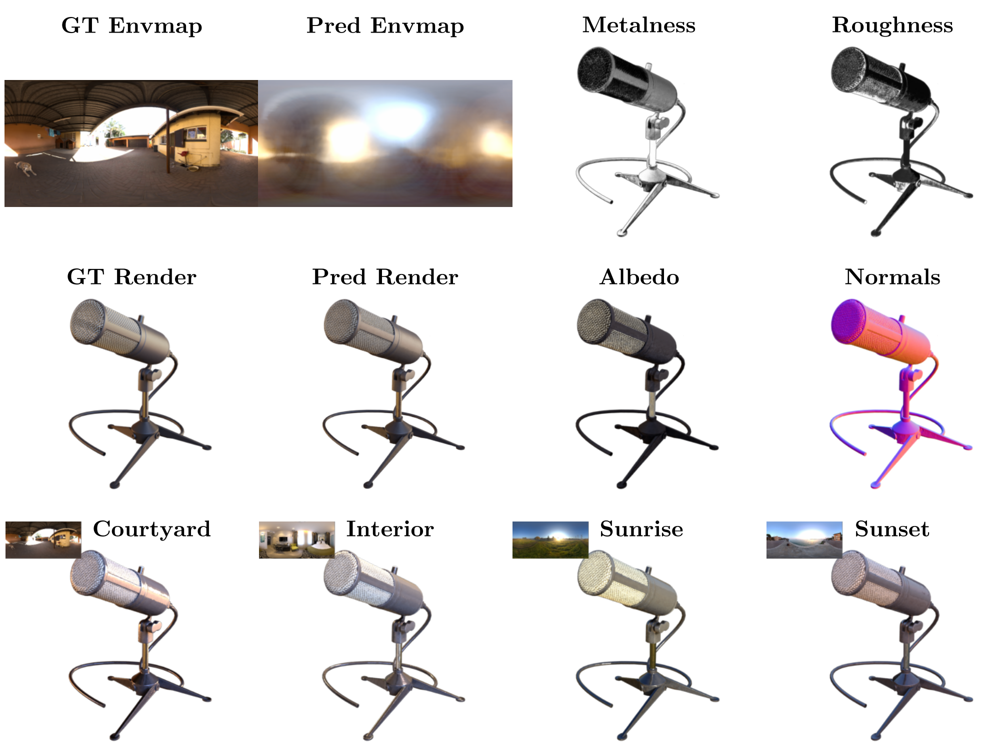

Figure 13: **Qualitative results on the Blender 'mic' scene.**

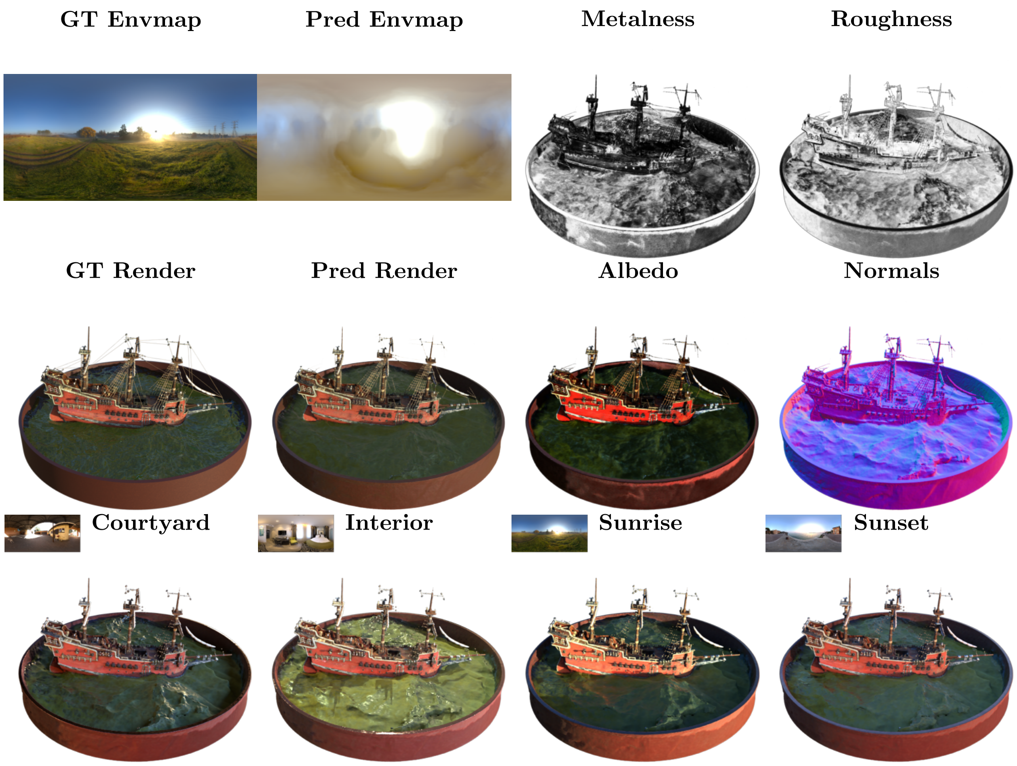

Figure 14: **Qualitative results on the Blender 'ship' scene.**

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

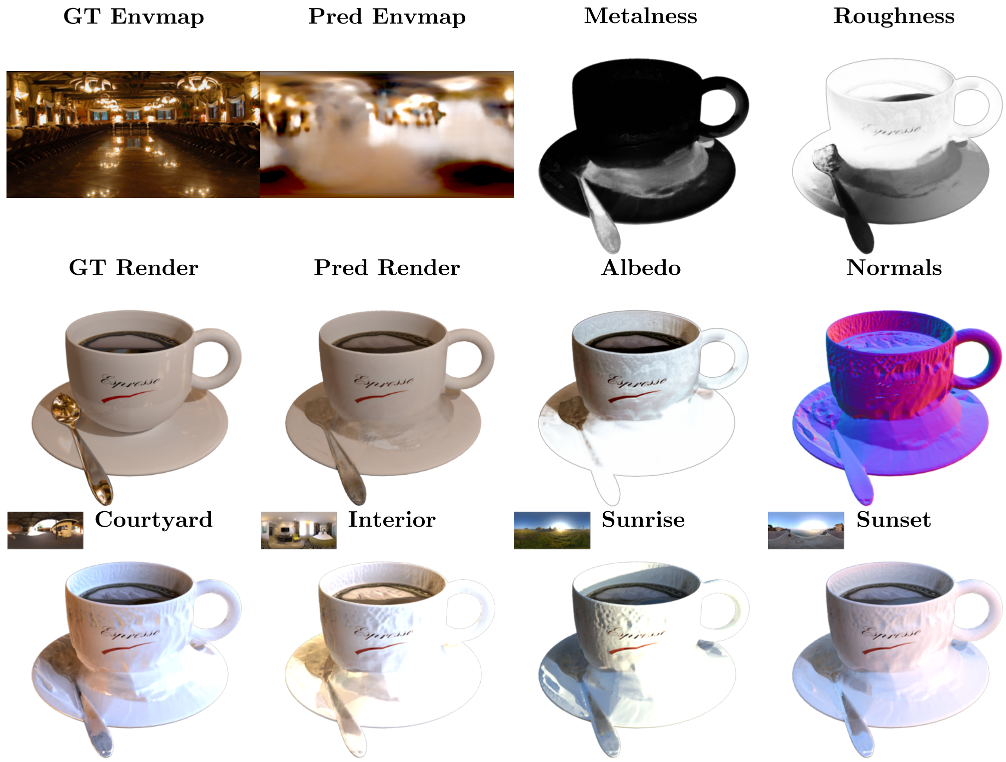

Figure 16: **Qualitative results on the Shiny Blender 'coffee' scene.**

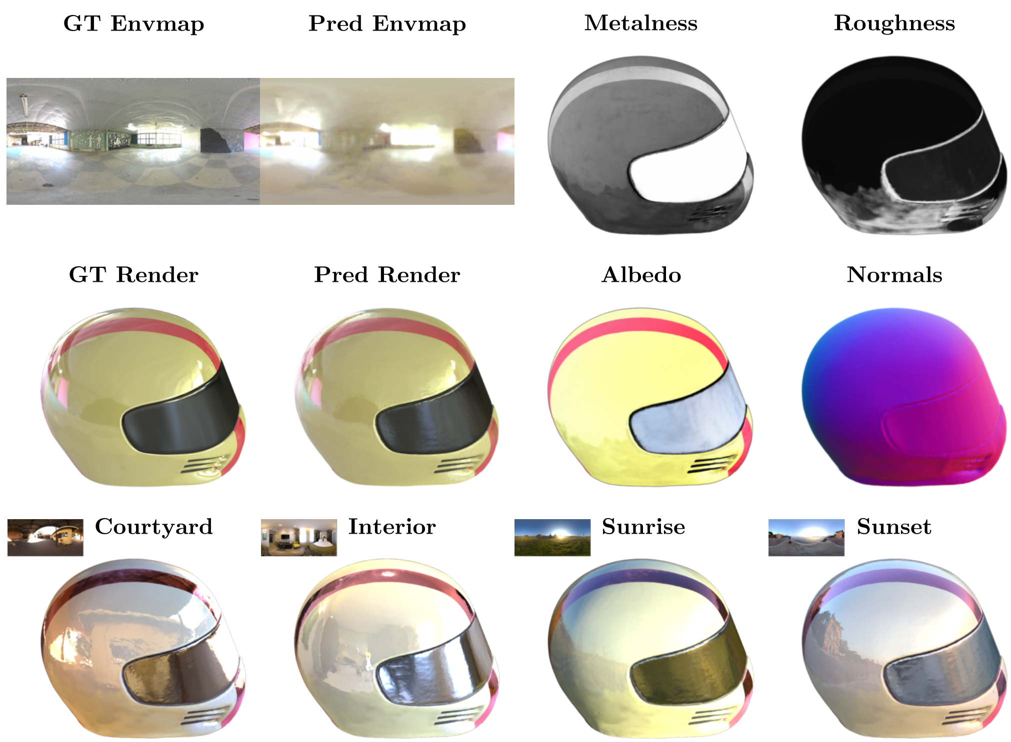

Figure 17: **Qualitative results on the Shiny Blender 'helmet' scene.**

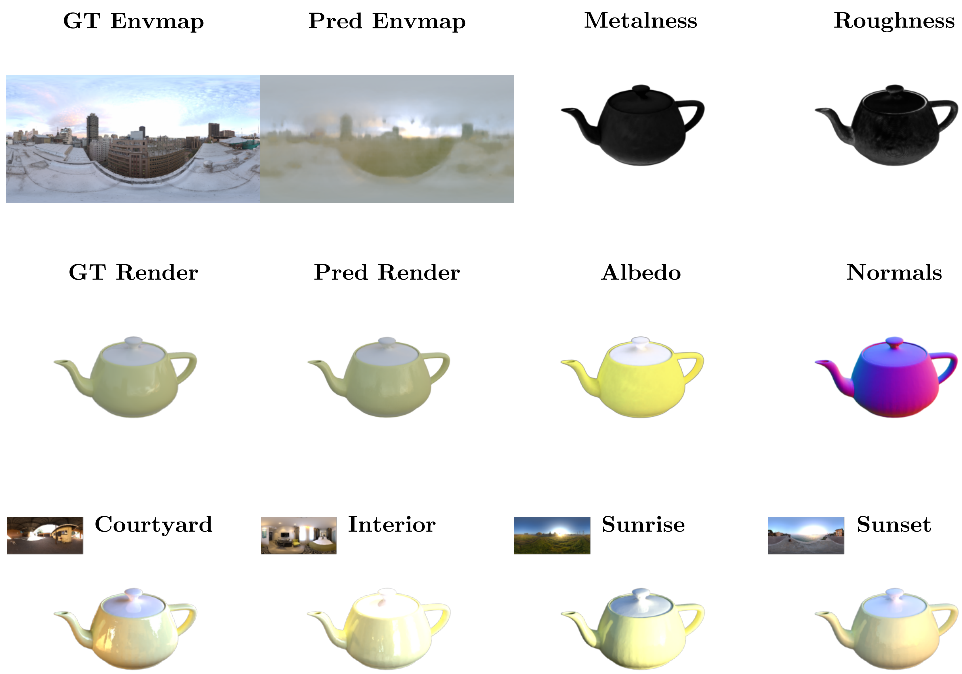

Figure 18: **Qualitative results on the Shiny Blender 'teapot' scene.**

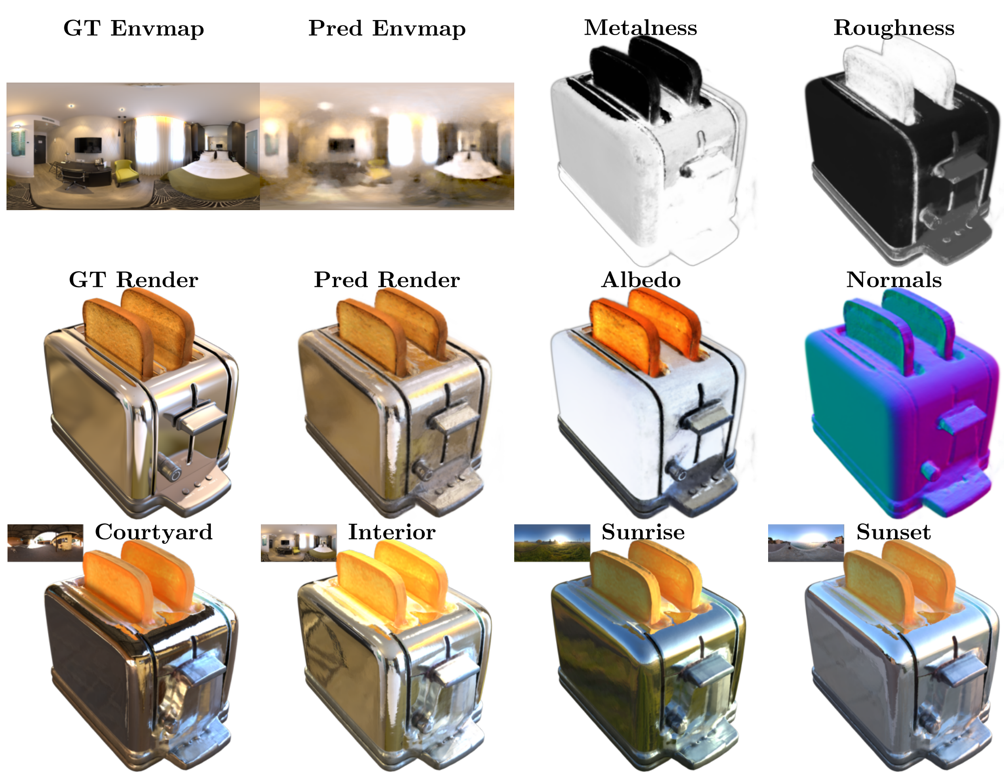

Figure 19: **Qualitative results on the Shiny Blender 'toaster' scene.**

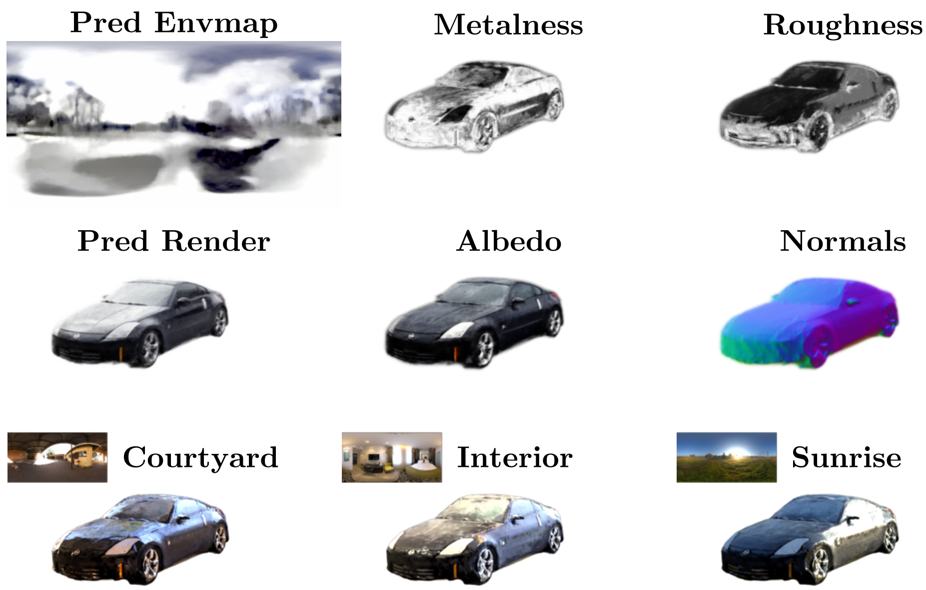

Figure 20: **Qualitative results on the CO3D car scene '421_58407_112553'.**

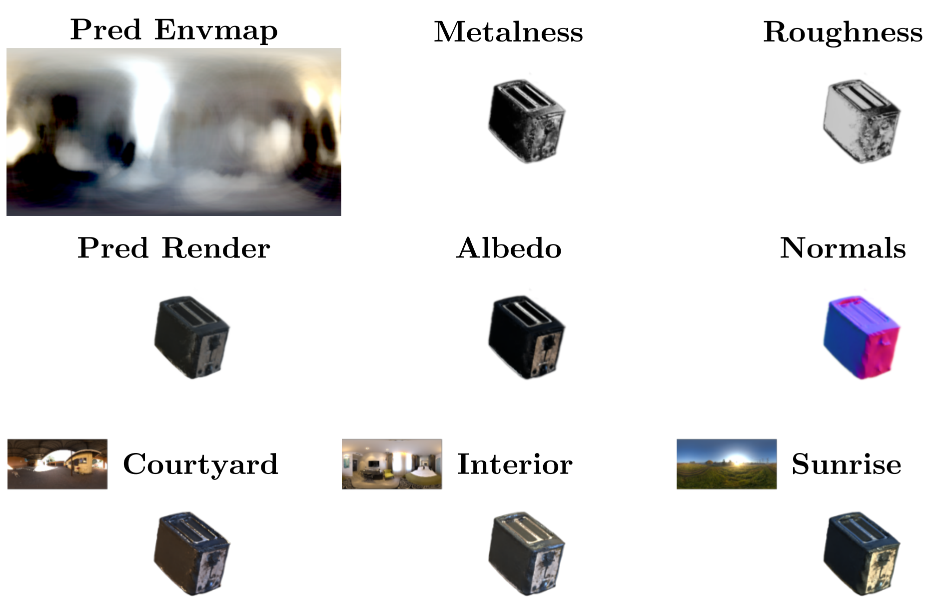

Figure 21: **Qualitative results on the CO3D car scene '112_13250_22955'.**

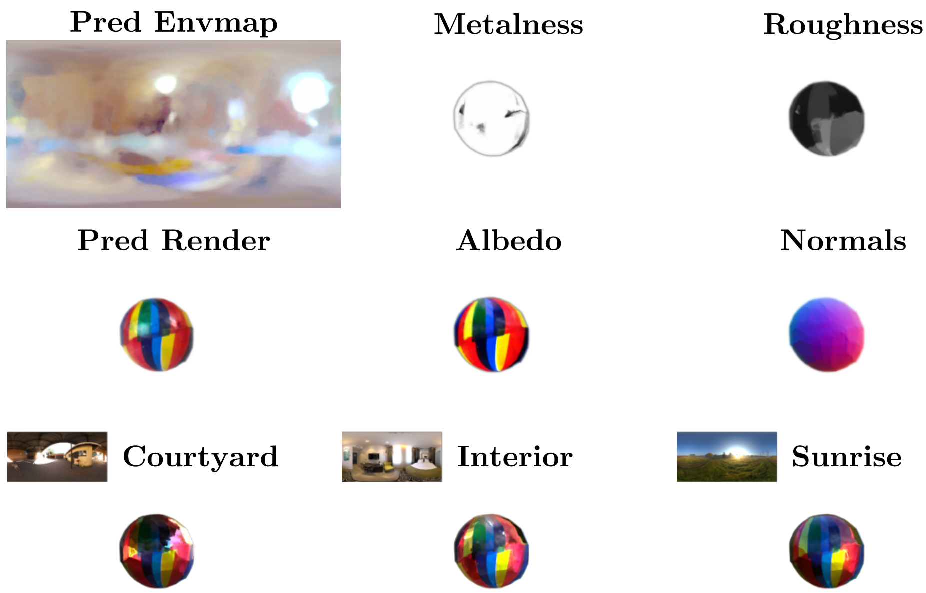

Figure 22: **Qualitative results on the CO3D ball scene '373_41665_83166'.**

Figure 23: **Qualitative results on the CO3D cup scene '34_1428_4472'.**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Claims made in the abstract and introduction are backed by experimental results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a discussion of the limitations of our work in section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the derivation along as assumptions made for our proposed illumination regularization and occlusion factors in appendices A.1 and A.3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the implementation details in appendix A.4. Additionally, we provide the code as supplementary material and will make the code public upon acceptance. All datasets used are publicly available.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is provided as a zip file along with the supplementary material. All datasets used are publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the implementation details in appendix A.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not provide error bars due to the runtime cost. While training for a scene is relatively fast for our method (1 A100 GPU hour), the total runtime including training and evaluation for all baselines in our main experiment becomes prohibitively expensive for computing error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the average compute time and computational resources used for each experiment. **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential positive societal impacts in section 1.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risk.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly credit the owners of data used in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.