Neural Ray-Tracing: Learning Surfaces and Reflectance for Relighting and View Synthesis

Julian Knodt[1] Seung-Hwan Baek[1] Felix Heide[1,2]
[1]Princeton University [2]Algolux

Abstract

Recent neural rendering methods have demonstrated accurate view interpolation by predicting volumetric density and color with a neural network. Although such volumetric representations can be supervised on static and dynamic scenes, existing methods implicitly bake the complete scene light transport into a single neural network for a given scene, including surface modeling, bidirectional scattering distribution functions, and indirect lighting effects. In contrast to traditional rendering pipelines, this prohibits editing surface reflectance, illumination, or composing other objects in the scene.

In this work, we explicitly model the light transport between scene surfaces and we rely on traditional integration schemes and the rendering equation to reconstruct a scene. The proposed method allows reflectance recovery with unknown light conditions and classic light transports such as pathtracing. By learning decomposed transport with surface representations established in conventional rendering methods, the method naturally facilitates editing shape, reflectance, lighting and scene composition. The method outperforms existing neural rendering methods for relighting under known lighting conditions, and produces realistic reconstructions for relit and edited scenes. We validate the proposed approach for scene editing, relighting and reflectance estimation learned from synthetic and captured views on existing datasets.

1. Introduction

View synthesis and scene reconstruction from a series of images is a fundamental problem in computer vision and graphics. Recent advances in neural scene rendering [5, 18, 32, 17] have achieved photo-realistic view synthesis results by encoding the entire scene's light transport into a learned volumetric representation. In contrast, traditional rendering approaches physically model light transport that is decomposed into illumination, reflectance, and multi-path scene inter-reflections. However, while physically-based differentiable rendering methods [19, 33, 19] allow for fine-grained lighting calculations, these rendering methods have been restricted to far more constrained settings than neural rendering approaches, solving for a small set of parameters with known geometries and lighting. In this work, we present a method that achieves high expressivity optimization with physically-based transport, separating the learning of the scattering functions, surface, and lighting, while providing the model capacity of volumetric representations.

Previous works on implicit neural radiance and reflectance representations [18, 5, 12, 17, 29, 35] do not represent scene components individually, but rely directly on a continuous volumetric representation of scene radiance, allowing for high reconstruction accuracy when interpolating views from a few RGB images. Because of the implicit radiance representations that these works rely on, light transport, surface representations, and scattering interactions are entangled. This prevents any one component from being changed or modified without fundamentally altering the others, prohibiting relighting and texture modification. Recently, attempts to incorporate these textures and relighting into existing neural pipelines have been proposed [1, 2, 28]. While these approaches allow for relighting, they still rely on volumetric representations that prohibit fine-grained, physically-based decomposition of transport offered by conventional rendering approaches. Instead of further augmenting reflectance fields, we move in an alternative direction, from a physically-based rendering framework, adding neural networks into the middle of the rendering pipeline to better approximate textures and lighting, providing an understandable and learnable reconstruction. Through this ease of understanding, we are able to modify our representation to handle relighting, retexturing, and reshaping. As an added benefit, our work is more easily incorporated into existing pipelines as compared to many neural models.

We present an approach for scene reconstruction with signed-distance fields for surface representation, spatially-varying high-frequency scattering functions, and parametrized light models, rendered using physically-based

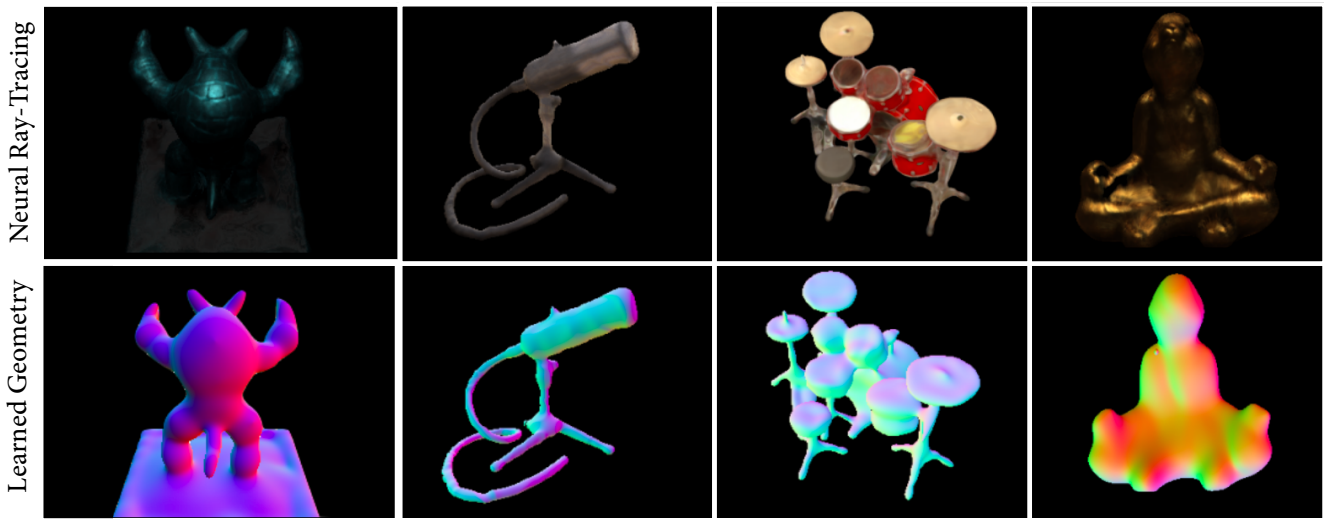

Figure 1: We demonstrate the ability of our methodology to recover surfaces from a set of RGB images. The first row is the recovered surface with learned BSDFs and learned lighting, and the second row are the surface normals. We show our approach on the NeRF [18] datasets, and the DTU [9] dataset. Instead of using a volumetric representation like NeRF, we use a surface-based representation with learned BSDFs in order to reconstruct scenes. We use a traditional ray-tracing and integration approach, with an entirely differentiable pipeline to find a solution to the inverse-rendering equation.

integration and sphere-marching during evaluation, only requiring extra raymarching for training. This partitions the optimization space into the discrete problem of surface and normal estimation, and texture and lighting estimation, allowing for interchanging of different components. We incorporate multiple multilayer perceptrons (MLP) inside of the rendering pipeline to mimic the universality demonstrated in volumetric neural representations [18, 1, 32], but in a principled fashion so as to maintain the understandability and efficiency of prior pipelines. We validate the proposed method using synthetic renderings and experimental data, where we outperform NeRV [28] and other approaches on their point-light dataset. Our code is available at `https://github.com/princeton-computational-imaging/neural_raytracing`.

Specifically, we make the following contributions in this work:

- An end-to-end surface-based forward and inverse rendering framework for scene reconstruction.
- We introduce a surface-based scene representation for shape and reflectance, that allows for multi-bounce transport.
- We demonstrate relighting and highly-editable learned representations, allowing for scene compositions, reflectance and surface editing.
- Validate the proposed method using synthetic data and experimental data, outperforming existing neural rendering methods that learn reflectance.

We note our method is not limited to RGB estimation, but can be used to directly estimate other surface properties, such as normals, as our method is not bound to the integration method.

2. Related Work

2.1. Neural Scene Representations

Monolithic Representations. With the advent of deep learning and neural networks, a growing body of work has explored methods for scene representations with convolutional neural networks [14, 21, 27] and implicit neural networks [16, 22, 18]. Notably, modeling the scene radiance with a neural implicit function, commonly in the form of a MLP, has proven to be effective, as demonstrated in NeRF [18]. MLP-based scene modeling integrates light radiance in a volumetric structure and optimizes the volumetric radiance function as a MLP that accurately reproduces a set of reference images, demonstrating learned view-dependant effects. While previous works are effective at novel view synthesis, most of them entangle reflectance, shape, and lighting in a single radiance volume, limiting its usage for physically-accurate scene decomposition [18, 32]. In contrast, we learn physically-based scene representations of shapes, reflectances, and lighting. This enables high-quality, efficient rendering with interpretability, and it allows for the same editability that conventional rendering pipelines offer.

Decomposed Neural Scene Representations. With the goal of extending radiance to reflectance, recent works have also explored decomposed modelling of reflectance [1, 28] and volumetric density to enable editable materials and lighting, similar to our approach. However, these approaches are restricted by the nature of volumetric rendering which is neither interpretable or editable, and makes multi-bounce rendering impractical, requiring 128 TPUs for training [28].

2.2. Differentiable Rendering with Ray-Tracing

Ray-tracing has been extensively studied with the goal of reproducing photorealistic imagery [23] using physically-based techniques. Inverting the rendering function gives us a framework known as inverse rendering, which is used to estimate scene compositions and parameters for a given set of input images. Recently, differentiable rendering has gained significant interest through frameworks implementing the forward rendering operations in a differentiable manner, so that backpropagation can be applied to optimize over some loss over the whole rendering pipeline [20, 19, 34]. While these efforts demonstrate promising results, this inversion faces great challenges. One of the central problems is the conventional representation of shape and reflectance. Shapes are commonly represented as a mesh in existing rendering approaches, imposing non-differentiable geometric discontinuity. While different approaches have been proposed to tackle this problem [7], it is still an open challenge to effectively and efficiently invert transport using existing mesh representations. Similarly, analytic reflectance models do not facilitate learning via gradient-based optimization, often due to discontinuities over lighting or low-likelihood of sampling specular highlights.

3. Inverting the Rendering Equation

The rendering equation has served as a firm ground for photo-realistic rendering that includes rich light transport details encompassing both direct and indirect lighting [11]. It describes the radiance coming from a 3D position $\mathbf{x}$ in the direction ω_o as

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega f(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n}) d\omega_i, \quad (1)$$

where $L_e(\mathbf{x}, \omega_o)$ and $L_i(\mathbf{x}, \omega_i)$ are the emitted and incident radiance for solid angle ω_o and ω_i respectively. Contributions from other scene points are integrated over the hemisphere Ω. f is the BRDF and $\mathbf{n}$ is the surface normals of the scene geometry [23].

Solving this rendering equation [23] requires two key ingredients: (1) representations of geometry, BSDF and lighting, and (2) an efficient integration scheme. In forward rendering, this is commonly solved with meshes, analytic reflectance and lighting models for the representations, and the integration problem is solved by pathtracing with importance sampling. Ray-tracing is commonly used for physically-based rendering, by directly modelling light bounces we are able to reconstruct realistic lighting conditions. While practical for forward rendering, this is problematic for inverse rendering as some operations are non-differentiable and these models may not be able to accurately model unknown conditions. For example, surfaces not directly illuminated are not able to be differentiated with respect to their spectrum in a direct rendering framework, and one-sided BSDFs with light hitting from the opposite side are also not differentiable.

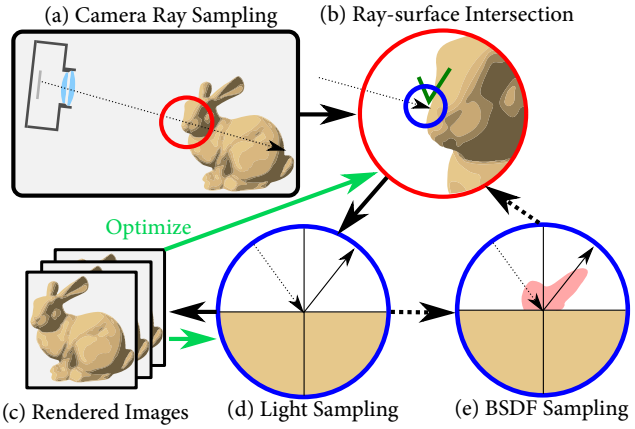

Figure 2: Overview of the proposed neural ray-tracing pipeline. For all integration schemes, we first take a cropped sample from UV space and transform it to camera space to compute rays from the camera center. Then, we perform sphere-marching to compute locations and normals of intersections. Given these positions, we perform light sampling in order to compute an incoming light direction and use the BSDF to compute an RGB value. If path-tracing, we sample a direction given the BSDF to compute a light-bounce ray and recurse for some given depth.

The proposed neural ray-tracing method aims to attain the rich light simulation properties of forward ray-tracing while resolving the non-differentiable nature of conventional representations by using fully-differentiable neural representations of surfaces and reflectances, while maintaining efficient light and surface intersection algorithms from traditional rendering. We then propose a practical method of solving the rendering equation following an analysis-by-synthesis principle using the proposed neural representations.

4. Neural Ray-Tracing

We present a ray-tracing framework for rendering and optimizing them based on analysis-by-synthesis principle,

and introduce a forward rendering function $\mathrm{Render}()$ from which the inverse rendering function $\mathrm{Render}^{-1}()$ is derived. Both functions are differentiable thanks to the neural shape and reflectance representations as described in later sections, enabling us to learn scene properties from photographic observations.

4.1. Forward Rendering

As in conventional pathtracing [23], we shoot a ray toward a camera pixel from the camera center. We then identify a surface point where the ray intersects with. With the shape as a neural SDF, we can efficiently implement this with sphere-tracing [6], see Figure 3. Once the surface point is located, we sample direct paths to light sources and multi-bounce paths using random sampling with the neural BSDF representations. By iterating with this intersection test, light and BSDF sampling we render an image captured by a camera, constituting our rendering function

$$I = \mathrm{Render}(\mathrm{SDF}, f, L, C), \tag{2}$$

where I is the rendered image, L is the lighting, and C is the camera, specified by its intrinsic and extrinsic parameters.

Lighting. When the lighting for a dataset is known, we model the lighting explicitly with known models, as in traditional ray-tracing. Otherwise, we assume static lighting conditions across images and model illumination directly as a light field with a MLP. This lighting MLP takes a spatial position, $\mathbf{x}$, and outputs the direction and intensity of incident light with primal contribution to that spatial position.

4.2. Inverse Rendering

Inverse rendering aims to invert the rendering function, $\mathrm{Render}^{-1}()$. In forward rendering ray-tracing, this involves evaluating numerous integrations and sampling with careful consideration of non-differentiable operations such as those meshes and reflectance. A key component for our work is that the forward rendering and neural scene representation are fully differentiable, and therefore they can be optimized by backpropagation. Based on an analysis-by-synthesis principle, our optimization function is defined as an iterative framework to find an optimal solutions for the shape SDF, reflectance f, and lighting C parameters that reproduce the input images. Specifically, the inverse function $\mathrm{Render}^{-1}()$ takes multiple images $\mathbf{I} = \{I_1, ...I_K\}$ as inputs captured at camera positions $\mathbf{C} = \{C_1, ...C_K\}$ and lighting configurations $\mathbf{L} = \{L_1, ...L_K\}$. K is the number of images. The inverse rendering function is then written as the optimization

$$\begin{aligned}
\mathcal{L}_p(I_{\text{ref}}, I) &= \frac{1}{3}(\|I_{\text{ref}} - I\|_1 + \|I_{\text{ref}} - I\|_2 + \|I_{\text{ref}} - I\|_2^{\frac{1}{2}}) \\
\mathcal{L}_s(M_{\text{ref}}, M) &= M_{\text{ref}} \log(M) + (1 - M_{\text{ref}}) \log(1 - M) \\
&= \arg\min_{\mathrm{SDF}, f} \sum_{i=1}^{K} \{\mathcal{L}_p\left(I_i, \mathrm{Render}\left(\mathrm{SDF}, f, L_i, C_i\right)\right) \\
&\qquad + \mathcal{L}_s\left(M_i, \mathrm{Silhouette}\left(\mathrm{SDF}, C_i\right)\right)\},
\end{aligned} \tag{3}$$

where $\mathcal{L}_p, \mathcal{L}_s$ are the photometric and geometric loss respectively. Specifically, the photometric loss we use is the sum of L1, L2, and RMSE losses between the input images I and the rendered images $\mathrm{Render}\left(\mathrm{SDF}, f, L_i, C_i\right)$. The geometric loss $\mathcal{L}_s$ is the binary cross-entropy loss between the ground-truth silhouette and the estimated shape silhouettes [32], where the silhouette map is defined as $\delta_{\text{surface visible}}$. To optimize over this silhouette map, we ray-march from each camera pixel through a scene and compute $\min_t \sigma(\mathrm{SDF}(r_t))$, the minimum SDF value along the ray. We use the AdamW optimizer [15] to optimize our parameters with respect to Equation (3). Our specific parameter set and learning rate are specified in the Supplemental Material. If the lighting is unknown, its MLP representation is also optimized in Equation (3).

5. Neural Shape and Reflectance Representations

We introduce our neural representations of shape and reflectance which are necessary for our differentiable scene representation. Recent progress on neural representations have fueled high-quality view synthesis [18] and reflectance acquisition [28, 1]. In this work, we devise novel neural shape and reflectance models suitable to be incorporated in ray-tracing where evaluation, modelling, intersection computation, and training efficiency are critical.

5.1. Two-level Shape Modeling

Recent methods use signed distance functions (SDF) [3] extensively as an implicit shape representation that can be differentiated for inverse rendering [30, 21]. SDFs takes a position in space, $\mathbf{x}$, and return the signed distance to the closest surface. The surface can be retrieved from a SDF as the zero level-set of the function. In order to *learn* a SDF, recent works show promising results of using a MLP [30, 21]. However, it is computationally expensive and unstable to use this MLP-based SDF representation in the ray-tracing pipeline. Therefore, we devise a two-level SDF representation consisting of spherical geometric sketches and detail refinement with a MLP as shown in Figure 3.

Geometric Sketches. We use spheres with parametrized radii and positions to describe low-frequency components

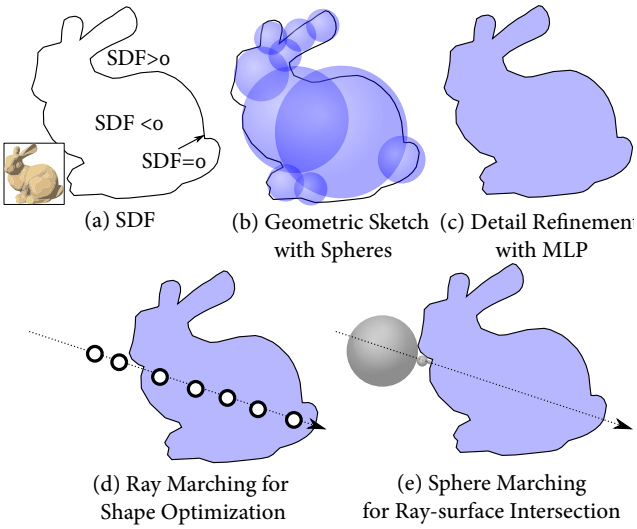

Figure 3: Our surface model: sphere-marching is used for computing surface intersections, and ray-marching is used for learning the surface representation. In order to inherit the flexibility of classical surface models and the universality of neural networks, our model is the sum of a MLP and a set of transformed spheres.

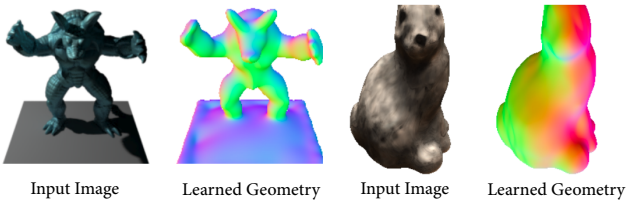

Figure 4: We visualize the learned surface through its surface normals. The Armadillo scene (left) has known point-source lighting condition while the lighting is unknown for the Rabbit scene (right). Scene data is from NeRV [28] and DTU [9, 32] respectively. We note that the top of the bunny is intentionally cut off in the sampled view.

of a true SDF. This is inspired by SDF based shape modelling as can be seen on Shadertoy [10]. The spheres are described as center and radius values, $\mathbf{c} \in \mathbb{R}^{N\times 3}$ and $\mathbf{r} \in \mathbb{R}^{N\times 1}$, where N is the number of spheres set as either $2^6, 2^7$ in our experiments, depending on the complexity of the object. We blend the basis sphere functions with smooth-min interpolation, resulting in the low-frequency SDF, $\mathrm{SDF}_{\mathrm{sketch}}$, as

$$\mathrm{SDF}_{\mathrm{sketch}}(\mathbf{x}) = -\frac{1}{k}\ln\sum_{i=1}^{N} e^{-ks(\mathbf{A}_i\mathbf{x},\mathbf{c}_i,\mathbf{r}_i)},$$
$$s(\mathbf{x},\mathbf{c},r) = \|\mathbf{x}-\mathbf{c}\|_2 - r, \tag{4}$$

where $s(\mathbf{x},\mathbf{c},r)$ measures the signed distance between a point $\mathbf{x}$ from a sphere surface and k is a smoothing parameter. Inspired by kernel regression, we define this SDF in kernel space for each sphere by a general transformation $\mathbf{A} \in \mathbb{R}^{3\times 3}$. This improves learning low frequency shapes, by adding more degrees of freedom to approximate. The benefit of our geometric sketch with spheres is their small number of learnable parameters ($\mathbf{A}$, $\mathbf{x}$, $\mathbf{c}$), facilitating efficient learning, crucial for our ray-tracing framework. This approach also guarantees the approximate correctness of our surface representation as a SDF, because it is known to be correct. However, high-frequency geometric detail cannot be effectively learned solely from this approach.

Geometric Refinement. To compensate for lost details in the sketch, we add geometric-refinement stage with an additional MLP. Specifically, we implement the geometric SDF residual using a MLP with softplus activation. Combining both geometric sketch and MLP-based refinement leads to an efficient and accurate SDF representation as

$$\mathrm{SDF}(\mathbf{x}) = \mathrm{SDF}_{\mathrm{sketch}}(\mathbf{x}) + \mathrm{MLP}(\mathbf{x}), \tag{5}$$

This SDF representation combines the power of explicit and implicit shape representations to enable efficient and effective learning for the differentiable ray-tracing rendering.

5.2. Neural Basis Reflectance Model

Reflectance modelling has been extensively studied for decades and its history goes back to the seminal analytic models of the Phong [24] model and the GGX model [31]. Recently, researchers have attempted to model BSDFs with neural networks for analysis [8] and inverse rendering [4]. However, this approach faces great challenges because of the complex spatial distribution of BSDFs over a limited set of views where per-point BSDF modelling becomes highly inefficient. To overcome this challenge in our ray-tracing framework, we combine recent MLP-based neural modeling and the concept of basis BSDFs and weight maps [13]. Representing both basis BSDFs and weight maps with MLPs allow us to model the spatially-varying BSDF of a scene in an efficient form for our ray-tracing framework, see Figure 5 for an overview.

To represent each basis BSDF b_i, we use an MLP that takes incident and exitant angles with respect to the surface normals and outputs trichromatic reflectance as

$$b_i(\phi_d,\theta_h,\theta_d) = \mathrm{MLP}(\phi_d,\cos\theta_h,\cos\theta_d)$$
$$= r_i, g_i, b_i, \tag{6}$$

where ϕ_d, θ_h, θ_d are the Rusinkiwewicz angles of azimuth, zenithal halfway angle, and incident halfway angle [26]. To avoid numerical instability and encode periodicity, we parameterize the MLP inputs by taking the cosine values for $\phi_d, \theta_h, \theta_d$. Given M basis BSDFs, we define their spatial distribution as a MLP-based weight function

$$w(\mathbf{x}) = \mathrm{softmax}\left(\mathrm{MLP}(\mathbf{x})\right) \in \mathbb{R}^{M\times 1}, \tag{7}$$

where $\mathrm{softmax}$ is the softmax function for normalization. Note that each basis BSDF is defined by one respective

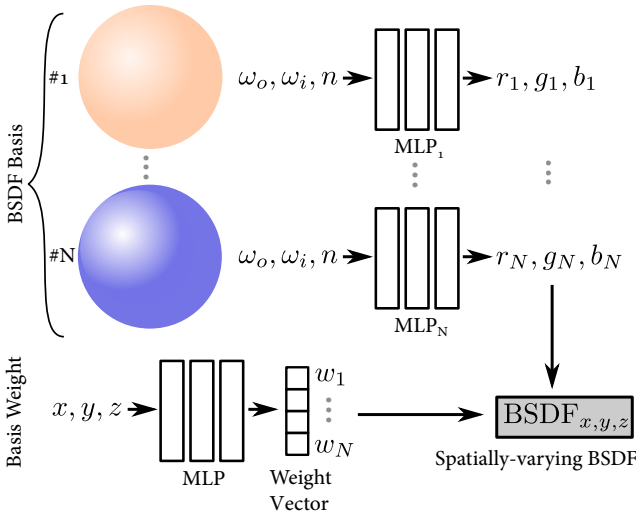

Figure 5: Illustration of our BSDF model. We take the dot product of a spatially-varying set of N weights with the spectrum output from a set of N component BSDFs.

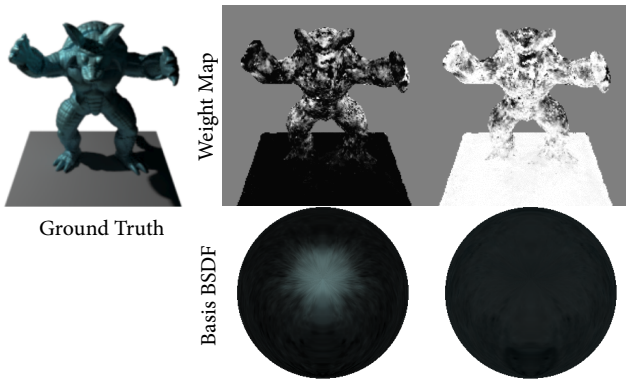

Figure 6: Our learned basis BSDF representations efficiently exploits spatial coherence in a scene, grouping surfaces with similar spectrums for efficient gathering of BSDF samples. We show two primal bases BSDFs and weights corresponding to the specular Armadillo and diffuse ground as well as their reflection.

MLP, the weight function over all bases is learned by a separate, single MLP to exploit spatial correlation of natural images. Combining the basis BSDFs and their corresponding weights, we obtain the complete BSDF representation as a linear combination

$$f(\mathbf{x}, \phi_d, \theta_h, \theta_d) = \sum_{i=1}^{M} w_i(\mathbf{x}) b_i(\phi_d, \theta_h, \theta_d), \qquad (8)$$

where w_i is the weight of the i-th BSDF. We are also feel free to add a classical model instead of a neural representation.

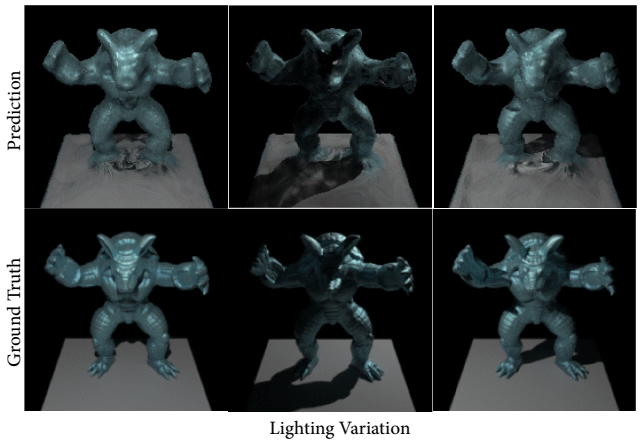

Figure 7: Our untangled representations of shape and reflectance allows us to perform effective relighting. By adding in learned occlusion, we are able to simulate soft-shadows cheaply, allowing us to learn shaded regions with cheap single-order bounces.

6. Assessment

We assess the proposed model by analyzing the light transport decompositions that our approach learns and applications of it, including relighting and the editability of the entire scene rendering pipeline. We refer the reader to the Supplemental Material for additional validation experiments that are not covered here due to limited space.

6.1. Neural Scene Representation

Geometry. We represent the shape as an implicit SDF with a sphere basis and a residual MLP, training on a set of RGB images with known segmentation masks, and corresponding camera positions. This allows for effective reconstruction of smooth shapes in a scene as shown in Figure 4. It is worth noting that we estimate accurate geometry under both known and unknown lighting conditions, since our model for lighting and surfaces are fully distinct.

Reflectance. Using the annotated RGB images, we reconstruct $K = 8$ basis BSDFs in the scene, as well as their weights at every point on the surface. We then qualitatively examine them to see if they appear physically plausible. Figure 6 shows that our method accurately reconstructs two components, which are approximately the specular and the diffuse BSDFs of the object.

6.2. Applications

Relighting. With the estimated geometry and reflectance, we are fully equipped to tackle the applications of relighting and view synthesis. Figure 7 shows our rendered images with three different point light source locations at a fixed viewpoint. Note that both hard shadows and soft shadows

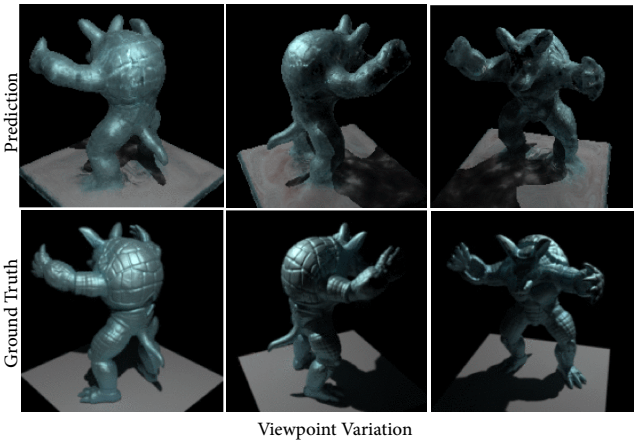

Figure 8: We perform free-viewpoint view synthesis leveraging our neural shape and reflectance representations. Our learned BSDFs and shadows from the learned surface are consistent across different viewpoints and maintain highlights.

are accurately predicted, demonstrating the accuracy of our representations for relighting.

View Synthesis. We also demonstrate view synthesis in Figure 8 by changing the viewpoints of the camera with fixed illumination, by varying it according to the test-set provided by NeRV [28]. Reflectances of the objects are consistent across the views and the shadow is faithfully reproduced because of our physically-based transport model.

Scene Editing. In order to demonstrate the editability of scenes and the rendering pipelines following our method, we explore performing various transformations of the BSDF, lighting and the surface, as shown in Figure 10. We explore these transformations on two datasets from NeRF [18], and one dataset from DTU [9]. Specifically, we showcase BSDF editing and geometry deformation, which consists of SDF transformations, and remapping portions of space to have a different composition of BSDFs, as can be seen in Fig. 10. We find implementing these changes to be close to that of a traditional forward renderer, and, since it is written in Python, is amenable to fast modifications, while being significantly faster than modifying a traditional renderer.

6.3. Quantitative and Qualitative Validation

We compare our method to recent neural reflectance methods including NeRV [28], Bi et al. [1], and a NeRF variant augmented with an illumination representation. Specifically, following [28], we compare against NeRF+LE, a NeRF variant, which combines the viewing direction with a light-embedding created with PointNet [25], to approximate varying light conditions. Figure 9 shows that our

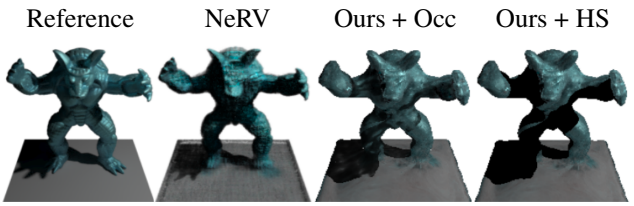

Figure 9: We compare our method on NeRV [28]'s public point-light datasets. Our method provides accurate RGB reproduction due to the disentanglement between the learned surface and BSDF. Adding hard shadows to our method, denoted as Ours+HS, enables geometry-based enhanced image contrast. Ours+Occ is a learned soft-shadow approximation, used during training to allow for differentiable approximations in regions of shadow, but can also be used at test time.

	Armadillo		Hotdog	
	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$	PSNR$^{\uparrow}$	MS-SSIM$^{\uparrow}$
NeRF+LE	20.35	0.883	19.96	**0.868**
Bi et al.	22.35	0.894	23.74	0.862
NeRV	22.14	0.897	23.93	0.863
Ours+Occ	**24.27**	**0.931**	**25.53**	0.860
Ours+HS	24.17	0.926	**25.53**	**0.868**

Table 1: We compare results against NeRV [28] and other methodologies on NeRV's single point-light dataset, with additional results in the Supplemental Document.

method outperforms existing methods both quantitatively and qualitatively, facilitated by our explicit surface model and smooth spatially-varying BSDF. By ensuring that only the BSDF captures relighting, the surface can be regularized to be smooth, whereas the other methods suffer from entangled surface and reflectance representations. Unfortunately, public code is not available for many recent methods, and, as such, it is necessary to report results on two public datasets adopted from [28]. Is is worth noting that Bi et al. [1] differs from our approach in that it requires a colocated point light source to learn. For training, NeRV [28] requires significant computing resources, e.g. 128 TPUs in their work whereas our method is memory-efficient training on a single 16GB GPU and can be generalized to various lighting configurations.

As the previous works of Bi et al. [1] and NeRV [28] do not yet provide public implementations, we compare two NeRF variants with illumination information on a new rendered datasets which has threes scenes including Plastic Bunny, Diffuse Teapot, and Plastic Buddha. Since we only train on a singular point light-source of fixed intensity and color, we compare against NeRF + PL, which directly embeds the 3-dimensional position of the point light into NeRF. In addition, we compare to NeRF + Env, which encodes the illumination along a fixed set of points along the scene's bounding sphere. Table 2 shows that our method outperforms the two variants.

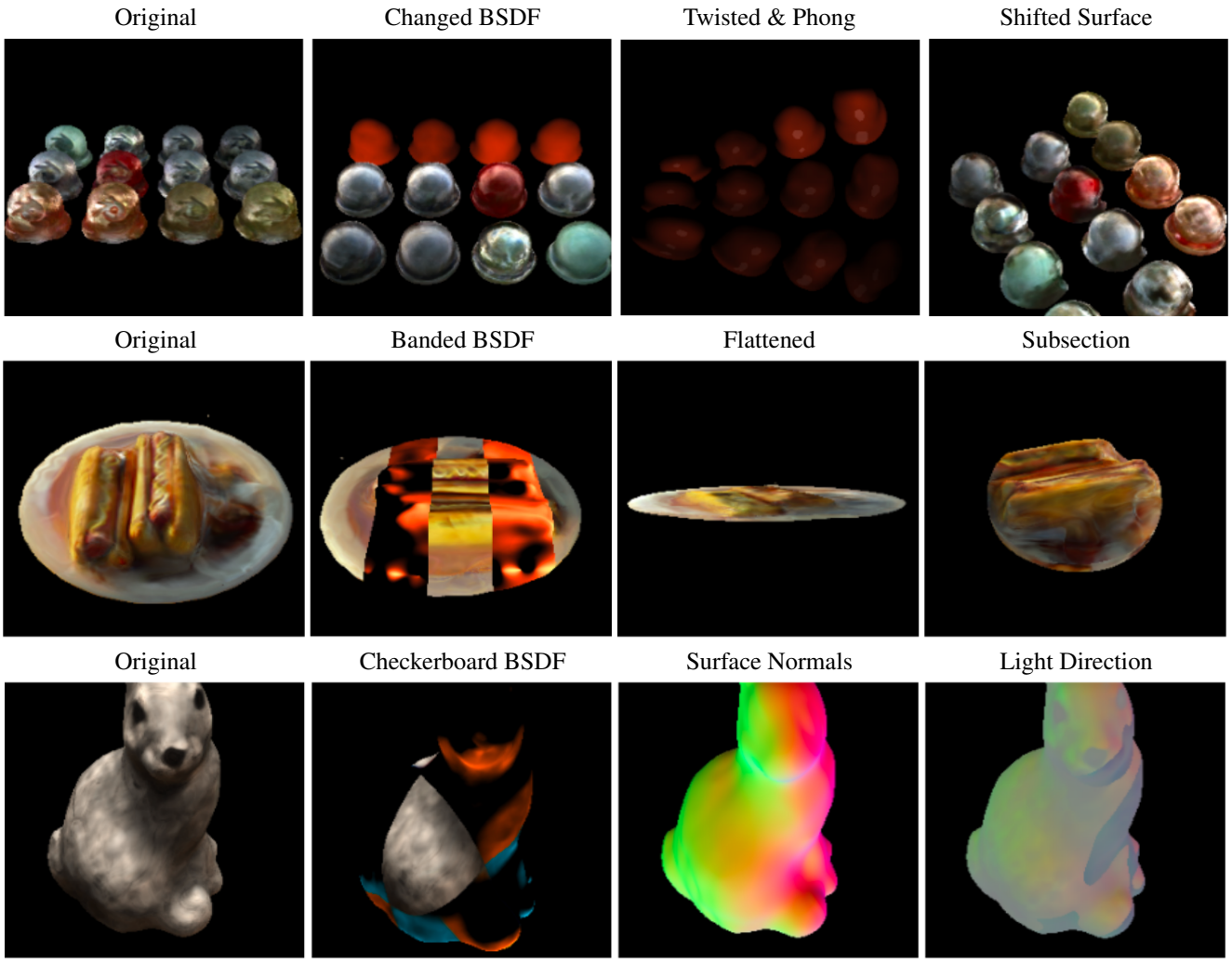

Figure 10: We demonstrate the editability of our methodology on two datasets from NeRF [18] and one from DTU [9]. To modify the surface representations, we perform SDF operations including intersection, subtraction, twisting, and translation of subsets of the surface. To modify the BSDF representation, we subsitute a diffuse red/orange BSDF over a subset of the image, such as adding bands, or changing the BSDF for all points with a positive z value. We also demonstrate the understandibility of our model, by showing surface normals and incoming light directions along the surface, by modifying integrators. All these changes took less than 15 LOC, and can be composited together to produce interesting effects. The learned BSDFs with unknown lighting may not be amenable to all transformations, due to unseen view positions far from the observed.

6.4. Multi-Bounce Pathtracing

We also perform multi-bounce pathtracing, to validate that it is possible with our methodology. We find that it is prohibitive for learning low-noise results, and that the cost incurred by random sampling outweights the benefits of having a soft-shadow. We refer to the Supplemental Material for more details.

7. Discussion

Our explicit surface modelling allows us to directly model the light transport, and thus efficiently compute light-samples and surface intersections, permitting liberal transformations, approximations of soft-shadows, and quickly learn representations. Due to the prohibitive cost and noise introduced through pathtracing, we explore an alternative to pathtracing for soft-shadow based illumination, and find it to be effective.

Comparing to prior work on view-synthesis and scene reconstruction, our work matches the performance of existing methods, while providing higher editability and physical interpretation thanks to our decomposed scene representations. Exploring more compact and easily-trainable decomposed scene representations is one extension which could be explored in future work.

Method	Plastic Bunny	Diffuse Teapot	Plastic Buddha
	PSNR$^{\uparrow}$		
NeRF+PL	12.92	11.82	12.11
NeRF+Env	12.59	11.60	13.12
Ours+HS	**16.58**	**17.96**	**15.53**
	SSIM$^{\uparrow}$		
NeRF+PL	0.738	0.708	0.670
NeRF+Env	0.743	0.700	0.652
Ours+HS	**0.746**	**0.777**	**0.736**

Table 2: We compare our method on a single-point relighting task to two NeRF variants, NeRF + PL and NeRF + Env. We note that since there are no inter-reflections in the scene, there are no soft-shadows and our hard-shadow model performs significantly better. See the text for a description of these compared methods, and the Supplement for additional benchmarks on these datasets.

8. Conclusion

Our work bridges the gap between recent MLP-based approaches for neural rendering which rely on a more complicated pipeline and traditional physically-based rendering approaches. We show that it is possible to combine concepts from conventional, physically-based, inverse-rendering methods with high-fidelity scene representations via neural networks. We hope this work motivates follow-up future work into physically-based rendering with embedded neural components where current theory cannot fill in the gaps. We further envision future neural rendering methods to depart from the volumetric models that is central to a large body of existing methods in neural rendering, making today's methods inefficient, uninterpretable, and challenging to integrate with the rich set of tools and methods developed in traditional rendering approaches – limitations that the proposed method makes steps towards lifting.

References

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

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

[3] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *ACM Siggraph Computer Graphics*, 24(2):109–116, 1990.

[4] Zhe Chen, Shohei Nobuhara, and Ko Nishino. Invertible neural brdf for object inverse rendering. In *Proc. of European Conference on Computer Vision (ECCV)*, Aug 2020.

[5] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent, 2019.

[6] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[7] Paul Henderson, Vagia Tsiminaki, and Christoph H. Lampert. Leveraging 2d data to learn textured 3d mesh generation, 2020.

[8] Bingyang Hu, Jie Guo, Yanjun Chen, Mengtian Li, and Yanwen Guo. Deepbrdf: A deep representation for manipulating measured brdf. *Computer Graphics Forum*, 39(2):157–166, 2020.

[9] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014.

[10] Pol Jeremias and Iñigo Quilez. Shadertoy: Live coding for reactive shaders. In *ACM SIGGRAPH 2013 Computer Animation Festival*, SIGGRAPH '13, page 1, New York, NY, USA, 2013. Association for Computing Machinery.

[11] James T. Kajiya. The rendering equation. In *Computer Graphics*, pages 143–150, 1986.

[12] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine, 2017.

[13] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics (TOG)*, 25(3):735–745, 2006.

[14] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[16] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[17] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[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, 2020.

[19] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: an adjoint method for lightning-fast differentiable rendering. *ACM Transactions on Graphics (TOG)*, 39(4):146–1, 2020.

[20] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), Dec 2019.

[21] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation.

In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[22] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *arXiv preprint arXiv:2003.04618*, 2, 2020.

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

[24] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, Jun 1975.

[25] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016. cite arxiv:1612.00593.

[26] Szymon M Rusinkiewicz. A new change of variables for efficient brdf representation. In *Eurographics Workshop on Rendering Techniques*, pages 11–22. Springer, 1998.

[27] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[28] 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. *arXiv preprint arXiv:2012.03927*, 2020.

[29] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 175–184, 2019.

[30] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. *arXiv preprint arXiv:2101.10994*, 2021.

[31] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. *Rendering techniques*, 2007:18th, 2007.

[32] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance, 2020.

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

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

[35] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.