

SPIDR: SDF-BASED NEURAL POINT FIELDS FOR ILLUMINATION AND DEFORMATION

Ruofan Liang^{1,4}, Jiahao Zhang¹, Haoda Li^{1,2}, Chen Yang³, Nandita Vijaykumar^{1,4}

¹University of Toronto, ²UC Berkeley, ³Shanghai Jiao Tong University, ⁴Vector Institute

¹{ruofan, nandita}@cs.toronto.edu ¹jiahaoz.zhang@mail.utoronto.ca
²haoda_li@berkeley.edu ³ycyangchen@sjtu.edu.cn

ABSTRACT

Implicit neural representations such as neural radiance fields (NeRFs) have recently emerged as a promising approach for 3D reconstruction and novel view synthesis. However, NeRF-based methods encode shape, reflectance, and illumination *implicitly* in their neural representations, and this makes it challenging for users to manipulate these properties in the rendered images explicitly. Existing approaches only enable limited editing of the scene and deformation of the geometry. Furthermore, no existing work enables accurate scene illumination after object deformation. In this work, we introduce SPIDR, a new hybrid neural SDF representation. SPIDR combines point cloud and neural implicit representations to enable the reconstruction of higher quality meshes and surfaces for object deformation and lighting estimation. To more accurately capture environment illumination for scene relighting, we propose a novel neural implicit model to learn environment light. To enable accurate illumination updates after deformation, we use the shadow mapping technique to efficiently approximate the light visibility updates caused by geometry editing. We demonstrate the effectiveness of SPIDR in enabling high quality geometry editing and deformation with accurate updates to the illumination of the scene. In comparison to prior work, we demonstrate significantly better rendering quality after deformation and lighting estimation.

1 INTRODUCTION

Recent advancements in implicit neural representations of 3D scenes, such as signed distance fields (SDF) (Park et al., 2019) and neural radiance fields (NeRF) (Mildenhall et al., 2020), have demonstrated exciting new capabilities in scene reconstruction and novel view synthesis. Recent research has investigated various aspects of implicit representations including training/inference acceleration, generative synthesis, modeling dynamic scenes, and relighting (Dellaert & Yen-Chen, 2020). An important and desirable aspect of 3D representations is their ability to be easily edited for applications such as augmented reality, virtual reality, 3D content production, games, and the movie industry. However, the geometry and illumination of objects/scenes represented by NeRFs are fundamentally challenging to edit. There are two major challenges in enabling the editability of neural representations. First, NeRF-based methods use a neural function with spatial coordinates as the input to predict the corresponding geometry and radiance information. Since the implicitly represented scene or object relies on the decoded information from every sampled spatial coordinate in the object’s modeling space, we cannot explicitly manipulate the geometry of the NeRF representation. Hybrid neural representations that also employ discrete representations (e.g., voxel grids, and hash tables) can improve geometry editability, but it is still limited to coarse-grained geometry deformation or composition. Second, high quality editing of scenes may require a corresponding change in the illumination of the object/scene. For example, unshadowed regions become shadowed if light occlusion happens after the deformation. Since illumination parameters are not explicitly modeled or represented in most NeRF-based methods, achieving accurate illumination after deformation is challenging.

Recent work that aim to improve the editability of NeRFs (Yuan et al., 2022; Bao et al., 2022) involve extracting mesh representations from the NeRF representations. However, these approaches do not

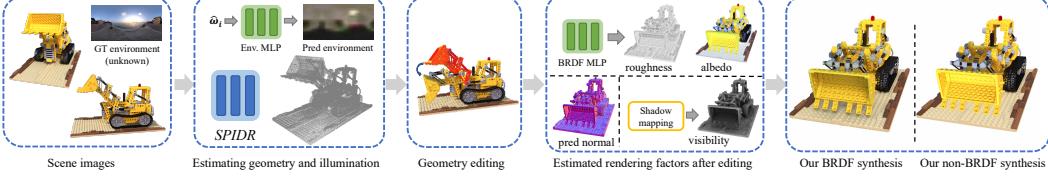


Figure 1: Given a set of scene images under an unknown illumination, SPIDR uses a hybrid neural implicit point representation to learn the scene geometry, radiance, and BRDF parameters. SPIDR also employs an MLP model to learn and represent environment illumination. After obtaining a trained SPIDR model, users can perform various geometry editing using our explicit point cloud representation. SPIDR then updates its estimated rendering factors based on user’s geometry editing. SPIDR finally uses these estimated rendering factors to synthesize the deformed object image using BRDF-based rendering.

tackle the scene illumination challenge posed by geometry deformation. Recent research has also investigated various aspects of illumination in NeRFs, including relighting scenes, material editing, and environmental light estimation. However, these approaches cannot be applied to illumination changes due to object deformations. Guo et al. (2020) enables relighting when composing scenes with multiple objects. This approach however does not enable shape editing nor the resulting change in illumination.

Our **goal** in this work is twofold. First, we aim to enable the fine-grained geometry deformation in NeRFs, including rigid body movement and non-rigid body deformation. Second, we aim to enable accurate illumination and relighting during object deformation and scene editing. To achieve this goal, we present SPIDR, an SDF-based hybrid NeRF model. SPIDR is designed based on two major ideas: **(1) SDF-based hybrid point cloud representation: enabling better editability.** We propose a hybrid model that is a combination of point cloud and neural implicit representations and is designed to accurately estimate the signed distance field (SDF). This hybrid representation enables direct manipulation of the object geometry by editing the point cloud or the mesh extracted from our representation for fine-grained rigid or non-rigid body transformation. Unlike prior work Xu et al. (2022), SPIDR estimates SDF which is critical to *(i)* extract higher quality meshes and surfaces for deformation and *(ii)* obtain better light visibility and surface normals for scene illumination. **(2) Accurate environment light and visibility estimation: enabling better illumination with deformation.** To estimate environment light more accurately, we employ an implicit coordinate-based MLP which takes in an incident light direction and predicts the corresponding light intensity. To enable illumination updates after deformation, we use the shadow mapping technique to efficiently estimate light visibility rather than a neural method as in prior work (Srinivasan et al., 2021; Zhang et al., 2021b). Shadow mapping approximates light visibility with estimated depth maps and is amenable towards deformation because it can accurately reflect the visibility updates caused by the deformation.

To summarize, our contributions are as follows:

- We introduce SPIDR, the first NeRF-based editing approach that enables geometry deformation, accurate illumination and shadow updates from deformation.
- We propose an SDF-based hybrid implicit representation with new regularizations that enables the reconstruction of high quality geometry for mesh extraction and illumination.
- We propose the use of a coordinate-based MLP to represent the environment light, which enables the accurate estimation of lighting that is required for relighting and illumination updates.
- We employ the shadow mapping technique to efficiently approximate visibility for the more accurate rendering of shadowing effects after the object deformation.

Our experiments demonstrate that SPIDR can obtain higher quality reconstructed object surfaces and more accurate estimations of the environment light. These essential improvements enable SPIDR to render scenes with deformed shapes while maintaining accurate illumination and shadowing effects.

2 RELATED WORK

Neural rendering and scene representation. Neural rendering is a class of reconstruction and rendering approaches that use deep networks to learn complex mappings from captured images to novel images (Tewari et al., 2020). Neural radiance field (NeRF) (Mildenhall et al., 2020) is one representative work that shows how the current state-of-the-art neural rendering methods (Barron et al., 2021; 2022; Verbin et al., 2021) combine volume rendering and neural network based implicit

representation for photo-realistic novel view synthesis. Follow-up works further improve the quality of reconstructed geometry (Yariv et al., 2021; Wang et al., 2021) using SDFs. However, SDF-based approaches have not been used in hybrid architecture or for scene editing in prior works. In order to make the training and rendering more efficient, hybrid neural representations that combine implicit MLP and other discrete spatial representations such as voxel grids (Liu et al., 2020; Fridovich-Keil et al., 2022), point clouds (Ost et al., 2022; Xu et al., 2022), and hash tables (Müller et al., 2022) have been proposed. These approaches however do not enable high quality scene editing and scene illumination after deformation.

Geometry editing and deformation. A large body of prior research investigate geometry modeling and editing for explicit 3D representations (e.g., meshes) (Kholgade et al., 2014; Jacobson et al., 2012; Yifan et al., 2020). These approaches cannot be directly applied to neural implicit representations. For neural representations, scene composition has been investigated by several works (Yang et al., 2021; Tang et al., 2022; Lazova et al., 2022), where objects can be added to or moved in the scene, but the shape of objects cannot be changed. Recent works also explore ways to achieve user-defined geometry deformation. Liu et al. (2021) allows shape editing with the user’s scribble but is limited to simple objects belonging to certain categories. Closest to our approach are Yuan et al. (2022); Bao et al. (2022), where they achieve more flexible object deformations by using mesh-guided deformation. Different from their methods, we use point clouds as explicit representation, which enables users also use point cloud for geometry editing. These works also do not address the illumination challenges with deformation. Guo et al. (2020) enables relighting when composing scenes with multiple objects. This approach however does not enable shape editing nor the resulting change in illumination.

Lighting estimation. Lighting estimation, also known as inverse rendering (Marschner, 1998), is a long-existing problem that aims to estimate surface reflectance properties and lighting conditions from images. Prior works have shown how to obtain accurate BRDF properties under known lighting conditions (Matusik, 2003; Aittala et al., 2016; Deschaintre et al., 2018); and how to estimate environment light from objects with known geometry (Richter-Trummer et al., 2016; LeGendre et al., 2019; Park et al., 2020). Recent works also leverage NeRF method to learn scene lighting and material reflectance. Bi et al. (2020); Srinivasan et al. (2021) attempt to learn the material reflectance properties with NeRF, but both methods require known lighting conditions. Boss et al. (2021a;b); Zhang et al. (2021a) jointly estimate environment light and reflectance with images under unknown lighting conditions, but their lighting approximation methods (e.g., spherical Gaussian, pre-integrated maps) do not consider self-occlusions, thus cannot render shadowing effects caused by occlusion Srinivasan et al. (2021); Zhang et al. (2021b) have explicit light visibility estimation in their model, allowing them to render relightable shadowing effects. However, the rendered results of these methods are over-smoothed and lack high-frequency details (demonstrated in Section 7.2). Additionally, these NeRF-based lighting estimation methods mainly focus on relighting static scenes, and none of them can be applied to address the illumination changes caused by geometry editing. In contrast, our model is able to render updated lighting and shadowing effects after editing the geometry of a NeRF scene.

3 PRELIMINARIES

MVS and Point-based NeRF. Several works leverage multi-view stereo (MVS) methods to provide geometry and appearance priors for NeRF, enabling efficient training and inference (Chen et al., 2021; Lin et al., 2021; Xu et al., 2022). Point-NeRF(Xu et al., 2022) is a hybrid NeRF representation with the point cloud representing explicit geometry. The point cloud used by Point-NeRF is first initialized by MVS methods (e.g., Schönberger et al. (2016); Yao et al. (2018)), then the image features extracted by a 2D CNN model are projected to the points in the point cloud as the neural point features. This neural point cloud is formulated as $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{f}_i)\}$, where \mathbf{p}_i and \mathbf{f}_i denote point position and point feature, respectively. During the volume rendering, \mathcal{P} generates neural features for sampled query points. Given a query point \mathbf{x} along the ray with direction¹ $\hat{\mathbf{v}}$, its volume density $\sigma(\mathbf{x})$ is an inverse distance interpolation of density values of the neighboring neural points. The density of each neural point conditioned on query point \mathbf{x} is predicted using a PointNet-like MLP (Qi et al., 2017) model F followed by a spatial MLP T :

$$\sigma(\mathbf{x}) = \sum_i \frac{w_{\mathbf{p}_i}}{\sum_i w_{\mathbf{p}_i}} T(\mathbf{f}_{i,\mathbf{x}}), \quad \mathbf{f}_{i,\mathbf{x}} = F(\mathbf{f}_i, \mathbf{x} - \mathbf{p}_i) \quad (1)$$

¹The hat symbol $\hat{\cdot}$ in this paper only denotes unit directional vector.

where $w_{\mathbf{p}_i} = \|\mathbf{x} - \mathbf{p}_i\|^{-1}$, denoting the inverse distance. The view-dependent radiance \mathbf{c} is predicted by a radiance MLP R with interpolated point features and view direction $\hat{\mathbf{v}}$ as the input:

$$\mathbf{c}(\mathbf{x}) = R(\mathbf{f}_{\mathbf{x}}, \hat{\mathbf{v}}), \quad \mathbf{f}_{\mathbf{x}} = \sum_i \frac{w_{\mathbf{p}_i}}{\sum w_{\mathbf{p}_i}} \mathbf{f}_{i,\mathbf{x}} \quad (2)$$

Then the radiance of sampled points \mathbf{x}_t along the ray \mathbf{r} is accumulated following NeRF's volume rendering equation:

$$\mathbf{C}(\mathbf{r}) = \sum_t w_t \mathbf{c}(\mathbf{x}_t), \quad w_t = \exp\left(-\sum_{j=1}^{t-1} \sigma(\mathbf{x}_t) \delta_t\right) (1 - \exp(-\sigma(\mathbf{x}_t) \delta_t)) \quad (3)$$

where δ_t denotes the distance between adjacent sampled positions along the ray.

SDF-based NeRF. Recent works show that implicit surface representation can improve the quality of NeRF's reconstructed geometry (Oechsle et al., 2021; Yariv et al., 2021; Wang et al., 2021). Yariv et al. (2021) illustrate a concrete mathematical conversion of volume density $\sigma(\mathbf{x})$ from predicted SDF $d_{\Omega}(\mathbf{x})$ of the space $\Omega \subset \mathbb{R}^3$ where the target object locates.

$$\sigma(\mathbf{x}) = \frac{1}{\beta} \Psi_{\beta}(-d_{\Omega}(\mathbf{x})), \quad \text{where} \quad \Psi_{\beta}(s) = \begin{cases} \frac{1}{2} \exp\left(\frac{s}{\beta}\right) & \text{if } s \leq 0, \\ 1 - \frac{1}{2} \exp\left(-\frac{s}{\beta}\right) & \text{if } s > 0 \end{cases} \quad (4)$$

where β is a learnable parameter, and Ψ_{β} is the cumulative distribution function (CDF) of the Laplace distribution. Following this parameterization, we can get a high-quality smooth surface from the zero level-set of predicted SDF values, and surface normal can be computed as the gradient of the predicted SDF w.r.t. surface point \mathbf{x} , i.e., $\hat{\mathbf{n}}_{\mathbf{x}} = \frac{\nabla d_{\Omega}(\mathbf{x})}{\|\nabla d_{\Omega}(\mathbf{x})\|}$.

The rendering equation. Mathematically, the outgoing radiance of a surface point \mathbf{x} with normal $\hat{\mathbf{n}}$ from outgoing direction $\hat{\omega}_o$ can be described by the physically-based rendering equation (Kajiya, 1986):

$$L_o(\mathbf{x}, \hat{\omega}_o) = \int_{\Omega} L_i(\mathbf{x}, \hat{\omega}_i) f_r(\mathbf{x}, \hat{\omega}_i, \hat{\omega}_o) (\hat{\mathbf{n}} \cdot \hat{\omega}_i) d\hat{\omega}_i \quad (5)$$

where $\hat{\omega}_i$ denotes incoming light direction, Ω denotes the hemisphere centered at $\hat{\mathbf{n}}$, $L_i(\mathbf{x}, \hat{\omega}_i)$ is the incoming radiance of \mathbf{x} from $\hat{\omega}_i$, $f_r(\mathbf{x}, \hat{\omega}_i, \hat{\omega}_o)$ is the bidirectional reflectance distribution function (BRDF) that describes the portion of reflected radiance at direction $\hat{\omega}_o$ from direction $\hat{\omega}_i$.

4 SDF-BASED POINT-NERF

This section describes how our hybrid NeRF model is parameterized for learning scene geometry and decomposed radiance (diffuse and specular). Since our objectives rely on high-quality reconstructed geometry for operations such as mesh extraction and shadow mapping, the original NeRF's reconstructed geometry is not good enough for these operations. Inspired by the success of using SDF for volume rendering (Yariv et al., 2021; Wang et al., 2021), we adopt an SDF-based parameterization (Yariv et al., 2021) for our hybrid NeRF model. Since we use a discrete hybrid model instead of a continuous MLP model, extra regularizations on SDF predictions are required.

4.1 PARAMETERIZATION

The structure of our model generally follows the design of Xu et al. (2022) described in Sec. 3. To incorporate SDF-based NeRF rendering, we make the density MLP T to predict SDF value d instead of volume density σ . These implicit SDF values d can then be converted into density values using equation 4. The geometry estimation MLP in our model now becomes

$$d(\mathbf{x}) = \sum_i \frac{w_{\mathbf{p}_i}}{\sum w_{\mathbf{p}_i}} d_i, \quad d_i = T(\mathbf{f}_{i,\mathbf{x}}) \quad (6)$$

To predict radiance, our model outputs both view-independent diffuse color and view-dependent specular color. This decomposition makes it easier for the model to learn view-independent reflectance (Section 5). The diffuse color \mathbf{c}_d is obtained by feeding the interpolated point feature $\mathbf{f}_{\mathbf{x}}$ (Eqn. 2) to the diffuse color MLP $R_d(\mathbf{f}_{\mathbf{x}})$. Similarly, the specular color \mathbf{c}_s is obtained using the another MLP branch $R_s(\mathbf{f}_{\mathbf{x}}, \hat{\mathbf{v}}, \hat{\mathbf{n}}_{\mathbf{x}}, \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{\mathbf{x}})$. We add $\hat{\mathbf{n}}_{\mathbf{x}}$ and $\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{\mathbf{x}}$ as extra inputs to this view-dependent MLP R_s . The inner product $\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{\mathbf{x}}$ represents cosine value of the angle between normal and view

direction (or reflectance direction). $\hat{\mathbf{v}}$, $\hat{\mathbf{n}}_{\mathbf{x}}$ and $\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{\mathbf{x}}$ are necessary terms for calculating specular radiance with BRDF. We expect the output of MLP R_s to be an approximation of the integral in Eqn. 5 with the specular BRDF.

The final radiance $\mathbf{c}(\mathbf{x})$ of each sampled point \mathbf{x}_t along the ray that is used in Eqn. 3 then becomes

$$\mathbf{c}(\mathbf{x}) = R_d(\mathbf{f}_{\mathbf{x}}) + R_s(\mathbf{f}_{\mathbf{x}}, \hat{\mathbf{v}}, \hat{\mathbf{n}}_{\mathbf{x}}, \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{\mathbf{x}}) \quad (7)$$

4.2 NORMAL REPRESENTATION

Surface normals are frequently used in BRDF rendering, however, computing normals from an MLP’s gradient is a costly operation. Thus, we explicitly attach a normal vector $\hat{\mathbf{n}}'_{\mathbf{p}_i}$ to each neural point \mathbf{p}_i in the point cloud. These normal vectors are supervised by the normals from the MLP’s gradients during training. Similar to other per-point attributes, the normal $\hat{\mathbf{n}}'_{\mathbf{x}}$ of the sampled point \mathbf{x} is an interpolation of normal vectors of the neighboring points. A simple weighted L2 loss is applied between the interpolated normal $\hat{\mathbf{n}}'_{\mathbf{x}}$ and the computed normal $\hat{\mathbf{n}}_{\mathbf{x}}$ for query point \mathbf{x} :

$$\mathcal{L}_n = \sum_{\mathbf{x}} w_{\mathbf{x}} \|\hat{\mathbf{n}}_{\mathbf{x}} - \hat{\mathbf{n}}'_{\mathbf{x}}\|^2, \quad \hat{\mathbf{n}}'_{\mathbf{x}} = \sum_i \frac{w_{\mathbf{p}_i}}{\sum_i w_{\mathbf{p}_i}} \hat{\mathbf{n}}'_{\mathbf{p}_i} \quad (8)$$

where $w_{\mathbf{x}}$ is the alpha-compositing weights of sampled point \mathbf{x} along its ray \mathbf{r} , as shown in Eqn. 3. In addition to facilitating deformation, these explicitly predicted normals can effectively eliminate noise in the directly computed normals (Verbin et al., 2021). The normal vectors used in the view-dependent MLP R_s will also be replaced with these predicted normals in the inference and later training steps.

4.3 SDF REGULARIZATIONS

Unlike Oechsle et al. (2021); Yariv et al. (2021); Wang et al. (2021) that use a large continuous MLP to implicitly represent SDF values for the whole scene, our model has local neural features attached to the discrete point cloud representation. The discretization of implicit neural representation can cause the loss of continuity of the represented signals (Reiser et al., 2021; Fridovich-Keil et al., 2022). Specifically, these discrete neural representations can degrade the reconstructed geometry in two ways: (1) neural points that are inside and far away from the object surface may not get sufficient updates to predict the correct density values; (2) the normal directions of points aligned with the same surface may be disturbed by the texture color or shadows. To address these issues, we add two regularization terms to the predicted SDF values based on SDF’s geometric properties.

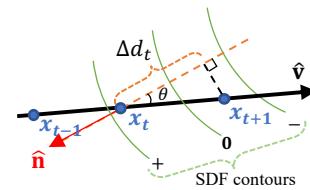
The first regularization is to stabilize the SDF prediction for points inside the surface. We want our model to be able to make stable negative SDF value predictions for the regions inside the surface. To achieve this goal, we propose a negative sparsity loss for all the sampled query points similar to Cauchy loss used in Hedman et al. (2021).

$$\mathcal{L}_s = \sum_{\mathbf{x}} \frac{1}{c} \log \left(1 + \frac{(1 - \beta \sigma(\mathbf{x}))^2}{c^2} \right) \quad (9)$$

where c is a hyperparameter that controls the loss scale, and β is the parameter used in Eqn. 4. By constraining the SDF value of the internal points, this regularization term can also prevent the model from incorrectly learning internal emitters (Verbin et al., 2021) instead of a solid surface when there are specular highlights.

The second regularization is to improve the continuity and consistency of predicted SDF values. Since the signed distance field defines the local spatial distance information of 3D positions, we can utilize this property to regularize the predicted SDF values along the sampled rays. Given two adjacent sampled points $\mathbf{x}_t, \mathbf{x}_{t+1}$ with sufficiently small distance along the same ray \mathbf{r}_t , the difference of predicted SDF values $\Delta d_t = d_{\mathbf{x}_{t+1}} - d_{\mathbf{x}_t}$ can be approximated by the projection of the relative distance $\delta_t = \|\mathbf{x}_{t+1} - \mathbf{x}_t\|$ in the normal direction $\hat{\mathbf{n}}_{\mathbf{x}_t}$, which is $\Delta d'_t \approx (\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{\mathbf{x}_t}) \delta_t$. We can therefore use the real distance value δ_t to constrain the changes of predicted SDF $d_{\mathbf{x}}$ with another L2 loss term

$$\mathcal{L}_d = \sum_t \|\Delta d'_t - \Delta d_t\|^2 \quad (10)$$



This regularization term can help our discrete NeRF model better learn the geometry of the target object without being affected by the discrete representation.

4.4 TRAINING

The training process is essentially the same as other NeRF methods, but with the above regularization terms. The full loss function is defined as

$$\mathcal{L} = \mathcal{L}_{color} + \lambda_n \mathcal{L}_n + \lambda_s \mathcal{L}_s + \lambda_d \mathcal{L}_d \quad (11)$$

where \mathcal{L}_{color} is an L1 loss between the ground truth and the rendered images, $\lambda_n, \lambda_s, \lambda_d$ are loss weight hyperparameters. In order to feed stable normal vectors to the specular MLP branch and to keep the specular MLP from being impacted by diffuse color, we exclusively train the diffuse MLP branch for a small portion of the total training steps at the beginning, then jointly train both diffuse and specular branches.

5 DECOMPOSING LIGHTING AND REFLECTANCE

In this section, we describe how we decompose the reflectance and environment light from the fully trained SDF-based PointNeRF model described in Sec. 4. There are three key components in our lighting estimation method. First, we use an HDR light probe image learned by a coordinate-based MLP to represent the environment light. Second, we employ the shadow mapping technique to approximate the surface light visibility. Third, we add extra MLP branches to estimate the surface BRDF properties. Our light estimation approach is designed assuming distant environment illumination and also does not consider indirect reflection.

5.1 NEURAL IMPLICIT ENVIRONMENT LIGHT

Surface visibility is a key factor in determining the rendering of hard cast shadows. However, environment illumination approximation methods such as spherical Gaussians (SG) (Zhang et al., 2021a; Boss et al., 2021a) cannot be integrated with surface visibility. Thus, in SPIDR, we first represent the environment light as an HDR light probe image (Debevec, 1998) (similar to Zhang et al. (2021b)). Every pixel on the light probe image can be treated as a distant light source on a sphere, thus the integral in the rendering equation (Eqn. 5) can be discretized as the summation over a hemisphere of the light sources (pixels) on the light probe image. To reduce computational complexity, environment light is estimated at a 16×32 resolution in the latitude-longitude format, which can be approximately treated as 512 point light source locations on a sphere centered around the object.

Unlike Zhang et al. (2021b) which directly optimizes 512 pixels \mathbf{p}_i as learnable parameters, we instead use a coordinate-based MLP E to implicitly represent the light probe image. Given the i th light source’s direction $\hat{\omega}_i$ (normalized coordinate on a unit sphere), MLP E takes $\hat{\omega}_i$ as input (with positional encoding (Mildenhall et al., 2020)) to predict the corresponding light intensity $E(\hat{\omega}_i)$. This parameterization enables SPIDR to learn high frequency lighting details while maintaining spatial continuity (Tancik et al., 2020). Since our environment light model represents an HDR image, we adopt an exponential activation function (Mildenhall et al., 2022) to convert the MLP output to the final light intensity.

5.2 COMPUTING VISIBILITY FROM DEPTH

Existing NeRF methods compute the surface light visibility via direct ray marching (Bi et al., 2020; Zhang et al., 2021b) or a visibility estimation MLP (Srinivasan et al., 2021; Zhang et al., 2021b). Although these methods can make accurate visibility predictions, they are either compute-intensive or are not applicable for geometry deformation. Therefore, we instead apply the shadow mapping technique (Williams, 1978) to approximate the light visibility. The shadow mapping method determines the light visibility by checking the depth difference between the depth map from the current viewpoint and the depth maps from the light sources under the same coordinate system. We thus utilize the estimated depth maps from our NeRF representation to indirectly calculate the light visibility of a point on the surface. Thus no additional computation is required during training. The visibilities of all light sources for point \mathbf{x} on the surface then form a visibility map $V(\mathbf{x}, \hat{\omega}_i)$ with the same shape as the light probe image. As a tradeoff for efficient training, the pre-computed depth maps do not allow for updates to the learned geometry.

5.3 NEURAL BRDF ESTIMATION

Similar to other prior works (Srinivasan et al., 2021; Boss et al., 2021b), we also use the microfacet BRDF (Walter et al., 2007), a commonly used analytic BRDF model, to describe the surface reflectance property. We use three additional MLP branches, with interpolation point feature f_x (Eqn.

²) as input², to predict BRDF parameters (diffuse $\mathbf{a} \in \mathbb{R}^3$, specular $\mathbf{s} \in \mathbb{R}^3$, and roughness $\alpha \in \mathbb{R}$). Recall that our NeRF model also decomposes radiance into diffuse color and specular color (Eqn. 7), which is in accordance with the BRDF decomposition. Our trained diffuse and specular MLP (R_d & R_s) can be served as priors for learning the corresponding BRDF parameters. We show how these priors are utilized as follows.

Diffuse Reflectance. Lambertian (diffuse) reflectance is mainly determined by the BRDF diffuse parameters $\mathbf{a}(\mathbf{x})$, also known as albedo. Since diffuse color can be perceptually treated as the shaded albedo, we directly initialize our albedo MLP branch $B_a(\mathbf{f}_x)$ with the weights from the diffuse MLP $R_d(\mathbf{f}_x)$. This provides a reasonable starting point for learning the real albedo.

Specular Reflectance. We use another two MLP branches, $B_s(\mathbf{f}_x)$ and $B_r(\mathbf{f}_x)$, to model the specular BRDF parameters specular \mathbf{s} (also known as Fresnel F0) and roughness α . Unlike the albedo branch $B_s(\mathbf{f}_x)$ that can be initialized with R_d , we cannot initialize these specular BRDF branches with our specular radiance MLP R_s , as R_s is conditioned on view directions and normals. Thus we need to optimize B_s and B_r from scratch. However, we still use the results from the specular radiance MLP R_s as one loss term to supervise the learning of specular BRDF.

5.4 TRAINING

During the training of lighting estimation, we jointly optimize environment light and BRDF parameters. In addition to using ground truth color \mathbf{C}^* for supervision, we also use diffuse color \mathbf{C}_d and specular color \mathbf{C}_s obtained from radiance MLP branches, R_d and R_s , to further constrain the BRDF integrated diffuse color \mathbf{C}'_d and specular color \mathbf{C}'_s respectively. Because we use pre-computed depth for visibility map during this training step, we freeze the first part of our model that outputs predicted SDF values $d(\mathbf{x})$ and radiance features \mathbf{f}_x , and only optimize these BRDF MLP branches (B_d , B_s and B_r) and our environment light model E . The loss function is formulated as

$$\mathcal{L}_{light} = \lambda_c \mathcal{L}_{color}(\mathbf{C}^*, \mathbf{C}'_d + \mathbf{C}'_s) + \lambda_l \mathcal{L}_{color}(\mathbf{C}_d, \mathbf{C}'_d) + \lambda_g \mathcal{L}_{color}(\mathbf{C}_s, \mathbf{C}'_s) \quad (12)$$

where λ_c , λ_l , and λ_g are loss weight hyperparameters. Since we already have strong geometry and radiance priors from the trained NeRF model, 10k iterations of training in this step are sufficient to get a reasonably accurate lighting estimation.

6 RENDERING DEFORMATION FROM STATIC SCENES

As described in Section 4, our hybrid NeRF model employs the point cloud as the explicit representation. Neural points in our final optimized neural point cloud are well aligned with the object surface. This enables users to easily select desired parts and perform common geometry transformations, including translation, rotation, and scaling. The local topology and neural point features are invariant to such transformations, thus we can render the deformed scene without retraining or updating any neural parameter. Figure 1 shows an example of the direct manipulation.

Since our SDF-based scene representation captures higher quality surfaces, we can utilize predicted SDF values or the point cloud itself to extract the object mesh. To achieve mesh-guided deformation, we register each neural point to its closest triangle face on the mesh. We compute the signed distance and the projected barycentric coordinate from the point projected onto the corresponding triangle face. After applying a deformation to the mesh (e.g., ARAP method (Sorkine & Alexa, 2007)), we apply a barycentric interpolation to find the new projected point locations and normals, and unproject the points along the normal by the signed distance.

7 EXPERIMENTS

We evaluate our method on various challenging 3D scenes. We make comparisons against prior works based on the quality of view synthesis, lighting estimation, and geometry deformation.

Datasets. We mainly use 2 two challenging datasets for evaluation: all the scenes in NeRF’s Blender synthetic dataset (Mildenhall et al., 2020) and real-captured scenes in BlendedMVS (Yao et al., 2020) that are used and processed by NSVF (Liu et al., 2020).

Baselines. We compare against different prior works for each evaluated task. For the novel view synthesis task, we choose NSVF (Liu et al., 2020), VolSDF (Yariv et al., 2021), and the original

²The final prediction of BRDF parameters is the weighted sum of predictions of sampled ray points.

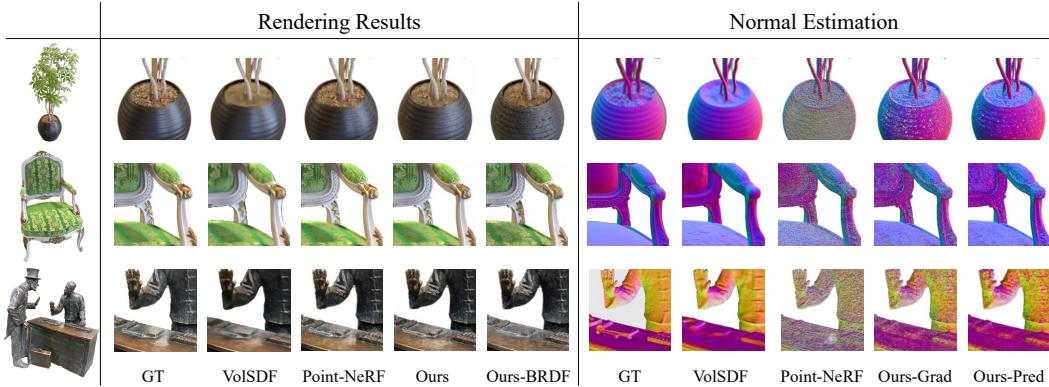


Figure 2: Qualitative comparison: rendering results on NeRF synthetic and BlendedMVS datasets.

Point-NeRF (Xu et al., 2022) as state-of-art baselines. For the lighting estimation task, we compare against NeRFactor (Zhang et al., 2021b) and Neural-PIL (Boss et al., 2021b). For the geometry editing task, we compare against the closest prior work, NeuMesh (Bao et al., 2022).

7.1 VIEW SYNTHESIS AND SURFACE RECONSTRUCTION

We use PSNR, SSIM, and LPIPS to evaluate the rendering quality. Additionally, we use MAE (mean angular error) to evaluate the estimated surface normals. Table 1 and Figure 2 show the quantitative and qualitative comparisons respectively. Note that Point-NeRF’s results are the per-scene optimized final results. Our SDF-based model without illumination decomposition (labeled “Ours”) generates rendering results comparable to the original Point-NeRF and better than NSVF and VolSDF. Thus, we conclude that our hybrid SDF-based model provides a high quality representation. For the original scenes without any editing, our model with light decomposition (“Ours-BRDF”), while providing significantly higher quality rendering under deformation, has slightly lower rendering quality compared to state-of-the-art static representations. This is because we currently only consider simplified direct illumination in our BRDF-based rendering and any surface points with incorrectly predicted normals or visibility maps can result in rendering artifacts (see the Ficus scene comparison, the first row in Fig. 2). SPIDR still however performs comparably to VolSDF.

Comparisons of estimated normals in Table 1 and Fig. 2 demonstrate the effectiveness of SDF-based presentations for learning scene geometry. Our SDF computed normals (“Ours-Grad”) and interpolated point normals (“Ours-Pred”) both have much better normal estimations than the original Point-NeRF. The interpolated normals filter out high frequency noise in the normals computed from SDF. VolSDF provides lower MAE, but the normals are typically over-smoothed. SPIDR is however able to capture more high frequency detail that lead to more realistic view synthesis.

7.2 LIGHTING ESTIMATION

We compare lighting estimation using re-rendered NeRF synthetic scenes provided by NeRFactor. Figure 3 shows the results for two selected scenes (Hotdog & Lego). Our model achieves significantly better rendering quality and geometry detail (see PSNR and MAE scores). Our estimated environment light for the Hotdog scene is significantly better than the two baseline methods. Unlike SPIDR, no existing work can accurately recover all light sources from the Hotdog scene. The environment light for the Lego scene is more challenging to infer as it does not have strong light sources. However, our SPIDR is still able to roughly estimate the light and dark regions. Unlike Zhang et al. (2021b); Boss et al. (2021b) that use additional material priors, our model directly estimates ma-

	Synthetic	BMVS
PSNR↑	NSVF	31.75
	VolSDF	27.96
	PointNeRF	33.31
	Ours	<u>32.30</u>
	Ours-BRDF	27.78
SSIM↑	NSVF	0.953
	VolSDF	0.932
	PointNeRF	0.978
	Ours	<u>0.976</u>
	Ours-BRDF	0.951
LPIPS↑	NSVF	0.047
	VolSDF	0.096
	PointNeRF	0.027
	Ours	<u>0.029</u>
	Ours-BRDF	0.061
MAE°→	VolSDF	19.87
	PointNeRF	48.09
	Ours-Grad	30.28
	Ours-Pred	<u>27.08</u>

Table 1: Quantitative comparison with baseline methods. BMVS represents NSVF’s BlendedMVS dataset.

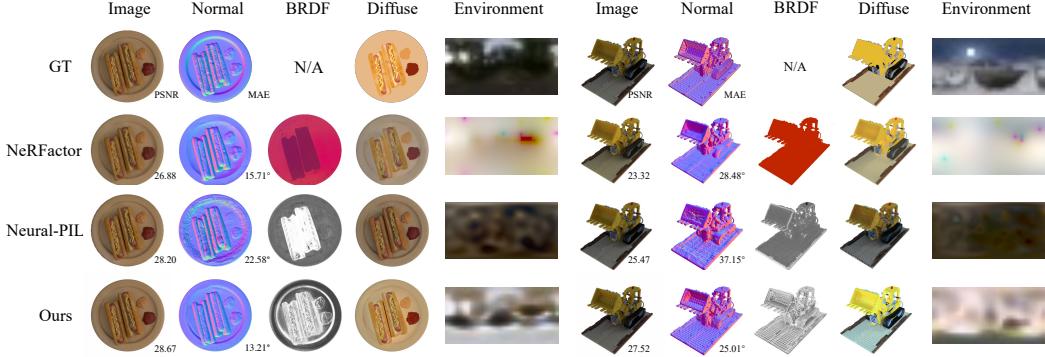


Figure 3: Lighting and BRDF estimation. In the BRDF column, NeRFactor visualizes the learned BRDF latent code, while Neural-PIL and our method show the grayscale material roughness values. In the Diffuse column, Neural-PIL shows rendered diffuse color, the other rows show the albedo colors.

terial properties from scene images without any material prior. Thus, SPIDR does not provide the highest quality BRDF estimation, but still successfully distinguishes different types of materials in the scene.

7.3 GEOMETRY EDITING

We qualitatively evaluate the geometry editing results by comparing against NeuMesh. We perform mesh-guided deformations on Blender synthetic scenes to achieve deformation effects similar to NeuMesh. Figure 4 shows the deformation results for two scenes (more results are in the Appendix). Our rendering results without lighting estimation have significantly better rendering quality with more texture detail compared to NeuMesh. Furthermore, our results with lighting estimation can successfully update the shadowing effects on the rendered image (see the shadow orientations on the side face of the deformed bulldozer blade). Our rendering results indicate that SPIDR can (1) successfully learn the environment illumination; (2) accurately update the light visibility for each point on the surface; and (3) effectively learn BRDF parameters that identify and remove existing surface shadows.

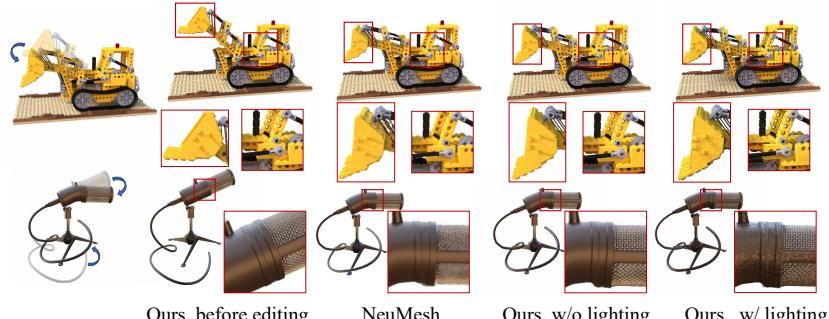


Figure 4: Qualitative comparison on geometry editing.

8 CONCLUSION

This paper introduces SPIDR, a new hybrid SDF-based NeRF architecture to enable more accurate and high quality scene editing and illumination. Our proposed neural representation with new SDF regularizations enables significantly better reconstruction of scene surfaces, while ensuring high quality view synthesis. Our new representation of environment light as a coordinate-based MLP estimates environment illumination more accurately compared to prior work. These contributions make SPIDR the first approach that can render scenes with fine-grained geometry deformations with the accurate resulting illumination. A current limitation of SPIDR is in its simplified BRDF-based rendering that only considers direct illumination; thus SPIDR is not able to capture shading effects caused by indirect illumination. Additionally, SPIDR’s SDF-based geometry representation may not work well for objects such as clouds or smoke. SPIDR however has great potential to be applied in various category-specific 3D generation and controlling tasks, such as controllable 3D human body/face and continuous scene synthesis for autonomous driving. We believe our proposed mechanisms can enable significant improvements in geometry editing and illumination for these tasks.

ACKNOWLEDGMENTS

We would like to thank NeuMesh’s authors for providing their deformation results. Resources used in this research were provided, in part, by the Vector Institute.

REFERENCES

- Miika Aittala, Timo Aila, and Jaakko Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (ToG)*, 35(4):1–13, 2016.
- Matthew Anderson, Ricardo Motta, Srinivasan Chandrasekar, and Michael Stokes. Proposal for a standard default color space for the internet—srgb. In *Color and imaging conference*, volume 1996, pp. 238–245. Society for Imaging Science and Technology, 1996.
- Chong Bao, Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision (ECCV)*, 2022.
- Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5855–5864, 2021.
- Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5470–5479, 2022.
- 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. *arXiv preprint arXiv:2008.03824*, 2020.
- Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12684–12694, 2021a.
- Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan Barron, and Hendrik Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. *Advances in Neural Information Processing Systems*, 34:10691–10704, 2021b.
- Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14124–14133, 2021.
- Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proc. SIGGRAPH 98*, pp. 189–198, 1998.
- Frank Dellaert and Lin Yen-Chen. Neural volume rendering: Nerf and beyond. *arXiv preprint arXiv:2101.05204*, 2020.
- Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)*, 37(4):1–15, 2018.
- Elmar Eisemann, Ulf Assarsson, Michael Schwarz, Michal Valient, and Michael Wimmer. Efficient real-time shadows. In *ACM SIGGRAPH 2013 Courses*, pp. 1–54. 2013.
- Christer Ericson. *Real-time collision detection*. Crc Press, 2004.
- Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinrong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5501–5510, 2022.

- Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*, 2020.
- Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5875–5884, 2021.
- Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. Fast automatic skinning transformations. *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012.
- James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pp. 143–150, 1986.
- Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.
- Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*, 2022.
- Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5918–5928, 2019.
- Haotong Lin, Sida Peng, Zhen Xu, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields with learned depth-guided sampling. *arXiv preprint arXiv:2112.01517*, 2021.
- Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.
- Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5773–5783, 2021.
- William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87*, pp. 163–169, New York, NY, USA, 1987. Association for Computing Machinery. ISBN 0897912276. doi: 10.1145/37401.37422. URL <https://doi.org/10.1145/37401.37422>.
- Stephen Robert Marschner. *Inverse rendering for computer graphics*. Cornell University, 1998.
- Wojciech Matusik. *A data-driven reflectance model*. PhD thesis, Massachusetts Institute of Technology, 2003.
- Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16190–16199, 2022.
- Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL <https://doi.org/10.1145/3528223.3530127>.

- Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5589–5599, 2021.
- Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18419–18429, 2022.
- 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.
- Jeong Joon Park, Aleksander Holynski, and Steven M Seitz. Seeing the world in a bag of chips. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1417–1427, 2020.
- Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.
- Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14335–14345, 2021.
- Thomas Richter-Trummer, Denis Kalkofen, Jinwoo Park, and Dieter Schmalstieg. Instant mixed reality lighting from casual scanning. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 27–36. IEEE, 2016.
- Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European conference on computer vision*, pp. 501–518. Springer, 2016.
- Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pp. 109–116, 2007.
- Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021.
- Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.
- Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. *arXiv preprint arXiv:2205.14870*, 2022.
- Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pp. 701–727. Wiley Online Library, 2020.
- Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. *arXiv preprint arXiv:2112.03907*, 2021.
- Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. *Rendering techniques*, 2007:18th, 2007.
- Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

- Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pp. 270–274, 1978.
- Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. *arXiv preprint arXiv:2201.08845*, 2022.
- Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13779–13788, 2021.
- Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *European Conference on Computer Vision (ECCV)*, 2018.
- Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1790–1799, 2020.
- Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.
- Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 75–83, 2020.
- Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: Geometry editing of neural radiance fields. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.
- Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.
- Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5453–5462, 2021a.
- Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.*, 40(6), dec 2021b. ISSN 0730-0301. doi: 10.1145/3478513.3480496. URL <https://doi.org/10.1145/3478513.3480496>.

A APPENDIX

A DETAILS OF LIGHT VISIBILITY AND BRDF-BASED RENDERING

A.1 LIGHT VISIBILITY

In section 5.2, we briefly demonstrate how we compute visibility maps from depth via the shadow mapping technique. We give more details in this section. Shadow mapping, a common method in computer graphics (Williams, 1978; Eisemann et al., 2013), aims to recover the per-point visibility maps from shadow maps of all light sources. The shadow maps are computed by comparing the depth difference between the depth map (z-buffer) from the view direction and the depth maps from the light sources after projecting to the same coordinate system.

Similar to other NeRF models, we use the expected termination point of each camera ray from NeRF’s ray marching to approximate the depth map D . Given the depth map D_c of the image captured by a camera with pose T_c , and the depth maps D_{ω_i} of light sources with pose T_{ω_i} , a pixel (u, v) that represents a surface point \mathbf{x} from the camera T_c has the shadow $S(u, v, \hat{\omega}_i)$ for the i th light source:

$$S(u, v, \hat{\omega}_i) = \mathbf{1}\{\text{interp}(D'_{\omega_i}, u, v) - D_c(u, v) + b\} \quad (13)$$

where $\mathbf{1}\{\cdot\}$ is a zero-one indicator function, $D'_{\omega_i} = T_c T_{\omega_i}^{-1} D_{\omega_i}$, representing the depth maps after projecting to T_c ’s coordinate system, b is the shadow bias term, $\text{interp}(\cdot)$ represents a bi-linear interpolation function. The visibility map V is a collection of values from shadow maps of all light sources $V(\mathbf{x}, \hat{\omega}_i) = S(u, v, \hat{\omega}_i)$. Since we treat our environment light as 512 light sources, 512 depth maps of the scene from each light source need to be pre-computed. To minimize this compute overhead, we render half-resolution light depth maps, which can be finished within 10 minutes for our tested scenes, but still provides reasonably accurate visibility maps from our observations.

A.2 NUMERICAL APPROXIMATION OF THE RENDERING EQUATION

Given light probe image E and visibility map V , we can approximate the discretized incoming radiance L_i with the product of light probe and its visibility, that is $L_i(\mathbf{x}, \hat{\omega}_i) = E(\hat{\omega}_i)V(\mathbf{x}, \hat{\omega}_i)$. The rendering equation (Eqn. 5) can thus be approximated as an accumulation of the reflectance from all discretized light directions:

$$\begin{aligned} L_o(\mathbf{x}, \hat{\omega}_o) &\approx \sum_{\hat{\omega}_i} L_i(\mathbf{x}, \hat{\omega}_i) B(\mathbf{x}, \hat{\omega}_i, \hat{\omega}_o)(\hat{\mathbf{n}} \cdot \hat{\omega}_i) \Delta \hat{\omega}_i \\ &= \sum_{\hat{\omega}_i} E(\hat{\omega}_i)V(\mathbf{x}, \hat{\omega}_i)B(\mathbf{x}, \hat{\omega}_i, \hat{\omega}_o)(\hat{\mathbf{n}} \cdot \hat{\omega}_i) \Delta \hat{\omega}_i \end{aligned} \quad (14)$$

where $\Delta \hat{\omega}_i$ denotes the solid angle for the corresponding incoming light direction $\hat{\omega}_i$ from the light probe image’s hemisphere.

B MESH-GUIDED POINT CLOUD DEFORMATION

In this section, we give more details on how we perform the mesh-guided point cloud manipulation discussed in Sec. 6 of the main paper. Extracting the guidance mesh from our neural representation is the first step for mesh-guided deformation. There are two approaches we can use to extract the mesh. The first approach is to perform the Marching Cube algorithm (Lorensen & Cline, 1987) to convert the zero-level set of the SDF estimated by our model into the triangle mesh. The second approach is to directly convert our model’s oriented point cloud representation to the mesh using point cloud-based mesh reconstruction methods such as Screened Poisson algorithm (Kazhdan & Hoppe, 2013). We interchangeably use both approaches to extract meshes. The Marching Cube method usually captures more accurate object surfaces, while the point cloud method can maintain more thin structures in the object. Given the extracted object mesh, we can perform any mesh-based deformation method to edit the object geometry. We mainly use ARAP (as-rigid-as-possible) method (Sorkine & Alexa, 2007) to perform the mesh deformations in this paper.

After obtaining the deformed mesh, the next step is to transfer the deformations made on the guidance mesh to the point cloud. The detailed procedure is described in Algorithm 1. Specifically,

Algorithm 1: Mesh Guided Deformation

```
Input : neural point cloud  $\mathcal{P}$ , extracted mesh  $(\mathcal{V}, \mathcal{F})$ , deformed mesh  $(\mathcal{V}', \mathcal{F}')$ 
1 for  $P_i$  in  $\mathcal{P}$  do
2    $F_i \leftarrow$  the nearest face to  $P_i$ ;
3    $\bar{P}_i \leftarrow$  the point projected from  $P_i$  onto  $F_i$ ;
4    $d_i \leftarrow \|P_i - \bar{P}_i\|$ ;
5    $\hat{n}_i \leftarrow (P_i - \bar{P}_i)/d_i$ ; /* normalized face normal */
6    $\alpha_i, \beta_i, \gamma_i \leftarrow$  the barycentric coordinate of  $\bar{P}_i$  on  $F_i$ ;
7 end
8  $\mathcal{P}' \leftarrow \{\}$ ;
9 for  $P_i$  in  $\mathcal{P}$  do
10   $V'_{0,i}, V'_{1,i}, V'_{2,i} \leftarrow$  the three corners of  $F'_i \in \mathcal{F}'$ ;
11   $\bar{P}'_i \leftarrow \alpha_i V'_{0,i} + \beta_i V'_{1,i} + \gamma_i V'_{2,i}$ ;
12   $P'_i \leftarrow \bar{P}'_i + d_i \hat{n}_i$ ;
13   $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{P'_i\}$ 
14 end
15 return  $\mathcal{P}'$ ;
```

we project each point in the point cloud onto its closest triangle face in the mesh and calculate the corresponding barycentric coordinate (Ericson, 2004) and point-to-face signed distance (Line 2-6 in Algorithm 1). These projected points on the triangle faces are moved in correspondence with the triangle faces. The positions of projected points after the deformation are determined by the exact barycentric coordinates calculated before (Line 10-11 in Algorithm 1). We then offset those projected points along the normals of deformed faces by the precomputed signed distances (Line 12 in Algorithm 1). Finally, these offset points form a new point cloud representing the deformed object (Line 15 in Algorithm 1).

We empirically observe that the distance between the point and the corresponding face is relatively small. Therefore, the mesh-to-point transfer will not result in significant artifacts. Compared to prior work (Yuan et al., 2022) that uses tetrahedrons to guide the deformation, our alignment mesh exhibits much simpler computations and enables more editing functionalities.

C IMPLEMENTATION DETAILS

Architecture and hyperparameters. We describe how our point-based neural implicit model is designed in this section. Figure 5 shows the overview of our model design. The feature and spatial MLP have 4 layers and 1 layer respectively. The conditioned neural feature $f_{i,x}$ has a feature size of 256. The radiance and BRDF MLP branches are 3-layer MLPs³ with 128 hidden units. Our environment light MLP is a simple 3-layer MLP with 128 hidden units. The relative position vector $p_i - x$, view direction \hat{v} , and incoming light direction $\hat{\omega}_i$ are encoded by the positional encoding (PE) (Mildenhall et al., 2020) (with PE level of 5, 4, 3, respectively), before feeding into MLPs. In addition, SPIDR’s final accumulated pixel colors are all converted to the sRGB by using a linear-to-sRGB tone mapping function (Anderson et al., 1996). Since MVS and point initialization step are not the main topics of this work, SPIDR remains the same model design as Point-NeRF (Xu et al., 2022) in the neural point cloud initialization step. In all our experiments, we use an MVSNet-based approach (Yao et al., 2018) to get the initial point cloud by default.

Training details After obtaining the initial neural point cloud via MVS approaches, we start training SPIDR on a single Nvidia RTX3090 GPU with Adam optimizer (Kingma & Ba, 2014) for both two training stages discussed in the main paper. We sample approximately 2048 rays per iteration for the training in both stages. Similar to Liu et al. (2020); Xu et al. (2022), we utilize the voxelized explicit representation to efficiently skip sampling points in the empty space. In the non-BRDF NeRF training stage (Sec. 4.4), we train our model for 160K iterations. We empirically set the loss hyperparameters $\lambda_n = 1e-3$, $\lambda_s = 2e-3$, $\lambda_d = 5e-3$ (λ_s also varies depending on the

³We use 5-layer specular radiance MLP for the Materials scene, as it contains more challenging metallic reflections.

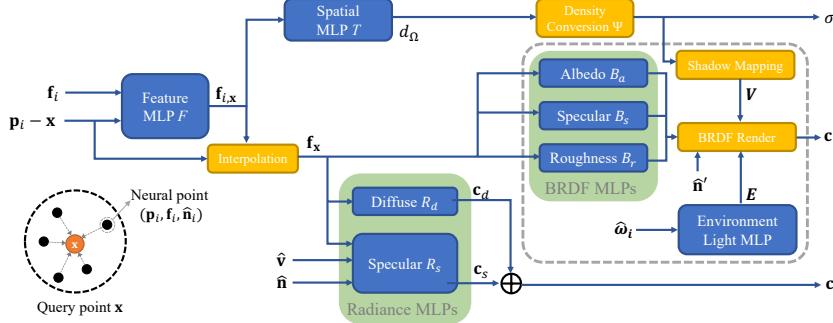


Figure 5: A visualization of SPIDR’s architecture. For simplicity, we omit the computation flow for both SDF computed normal $\hat{\mathbf{n}}$ and interpolated point normal $\hat{\mathbf{n}}$ (Eqn. 8).

specularity of the scene). We set the initial learning rate to 5e-4 for MLPs and 2e-3 for neural point features. In order to get relatively stable normals for the specular MLP branch R_s , we skip the training of R_s branch and treat the diffuse radiance as the final radiance during the first 10k training iterations. In the BRDF training stage (Sec. 5.4), we train the BRDF modules (BRDF MLP branches and environment light model) and environment light MLP for 10k iterations. We set the loss hyperparameters $\lambda_c = 0.5$, $\lambda_l = 1.0$, $\lambda_g = 1.0$. The initial learning rates for BRDF MLP branches and the environment light MLP are set to 5e-4 and 1e-3, respectively.

D POINT PRUNING AND GROWING

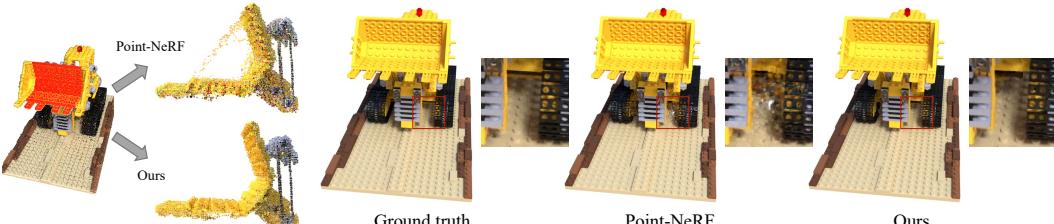
Although the original Point-NeRF (Xu et al., 2022) has its own point pruning/growing mechanisms and performs well in various scenes, it still has limitations that can impact the rendering quality. Since SPIDR utilizes the explicit point cloud representation to control the object deformation, our method relies on high quality point cloud representation to achieve fine-grained object deformations. The outlier points that are far away from the object surface can thus affect the quality of point cloud deformation, as it is hard to know which particular geometry region these outlier points are aligned with (e.g., outlier points above Lego bulldozer’s blade, as shown in Figure 6a). In addition, these outlier points floating above the object surface could cause rendering artifacts. For example, Figure 6b shows the original Point-NeRF’s rendering artifacts in a less observable region. Although we can observe a hole in the highlight region of Fig. 6b, Point-NeRF’s point growing mechanism cannot effectively fill this hole as the outlier points floating above the real surface become the fake floating volume. During the ray marching, each ray can hit the fake floating volume before hitting the real surface. Thus, Point-NeRF’s point growing mechanism is unable to be triggered to add points in this region. To improve the quality of point cloud representation, we purpose new point pruning and growing mechanisms for SPIDR.

D.1 POINT PRUNING

The original Point-NeRF uses a learnable point confidence attribute to prune the low-confidence points, however, this pruning method does not guarantee the outlier points can be completely removed (see Figure 6a). In SPIDR, we completely discard the point confidence attribute, and use estimated SDF values and recorded compositing weights to prune points. There are two key techniques in our pruning mechanism:

Pruning based on estimated SDF. We can easily compute per-point SDF values by treating a neural point’s position as a query point’s position (i.e., $\mathbf{x} = \mathbf{p}_i$). We then prune points with absolute SDF values larger than the pre-defined threshold (we use the SDF value s such that $\frac{1}{\beta}\Psi_\beta(-s) = 0.99$ as the threshold).

Pruning based on recorded compositing weights. The second pruning technique is based on an observation of compositing weights (Eqn. 3), that is, if a query point \mathbf{x}_t always has a very low compositing weight w_t given a camera ray from any direction, then the region where this point locates is likely to be an empty space. We utilize this fact to prune neural points that are always associated with query points with low compositing weights. Specifically, we keep a record of each neural point’s maximum compositing weight that their associated query points can achieve during a



(a) Point cloud after training (b) Final rendering results on the region where the initial points are sparse.

Figure 6: The comparison of point cloud pruning/growing on the Lego scene. Point-NeRF and our model have the same point cloud initialization before the NeRF training.

fixed number of training iterations. We then prune points with recorded weights lower than a pre-defined threshold (we set the threshold to 0.02). After the pruning, we clear the recorded weights and start a new recording for the next round of pruning.

We jointly use the above two pruning methods to remove outlier points and random points inside the object. We only run the pruning in the NeRF training stage (Sec. 4.4), and the pruning is performed once for every 10k training iterations. As demonstrated in Figure 6a, our purposed pruning mechanism can effectively remove the outliers that the original Point-NeRF is unable to remove.

D.2 POINT GROWING

In addition to the original Point-NeRF’s point growing mechanism, we also propose a new point growing mechanism in order to add missing points to the complicated scenes. Our new point growing mechanism works on the voxelized point cloud. Our method attempts to add new points to the neighbors of voxels that are very close to the surface (the estimated SDF values close to 0) but contain relatively few neighboring neural points. Please see Algorithm 2 for more detailed steps of our point growing.

Similar to the point pruning mechanism, point growing is only applied to the NeRF training stage. We run our purposed point growing once for every 40k training iterations. Figure 6b demonstrates the effectiveness of our purposed point growing mechanism. Compared to the Point-NeRF, our rendering results successfully recovers the missing points on the surface that is less observable.

Algorithm 2: Our Proposed Point Growing

```

Input : neural point cloud  $\mathcal{P}$ , SDF threshold  $T_{\text{SDF}}$ , point number threshold  $T_{\text{points}}$ ,  $K$ ,  

       MLP
Functions: neighbor( $V$ ): return a set of 26 adjacent neighboring voxels of voxel  $V$ .  

            num_points( $\mathcal{V}$ ): return the number of points contained by voxel set  $\mathcal{V}$ .  

            topk_min_voxel( $\mathcal{V}$ ,  $K$ ): return top  $K$  voxels with fewest points in the voxel set  

 $\mathcal{V}$ .
1 Voxelize point cloud  $\mathcal{P}$  to voxel set  $\mathcal{V} = \{V_i\}$ ;  

2 for  $V_i$  in  $\mathcal{V}$  do  

3    $s_i \leftarrow$  the estimated SDF of the voxel center  $c_i$ ;  

4    $\mathcal{V}_{\text{local}} \leftarrow \{V_i\} \cup \text{neighbor}(V_i)$ ;  

5    $n_i \leftarrow n_i + \text{num\_points}(\mathcal{V}_{\text{local}})$ ;  

6 end  

7  $\mathcal{V} \leftarrow \{V_i \mid n_i < T_{\text{points}} \text{ and } |s_i| < T_{\text{SDF}}\}$ ;  

8 for  $V_i$  in  $\mathcal{V}$  do  

9    $\mathcal{V}_{\text{local}} \leftarrow \{V_i\} \cup \text{neighbor}(V_i)$ ;  

10   $\mathcal{V}_{\text{sparse}} \leftarrow \text{topk\_min\_voxel}(\mathcal{V}_{\text{local}}, K)$ ;  

11   $\mathcal{P} \leftarrow \mathcal{P} \cup \{c_i \mid V_i \in \mathcal{V}_{\text{sparse}}\}$ ; /* Add voxel centers */
12 end  

13 return  $\mathcal{P}$ ;

```

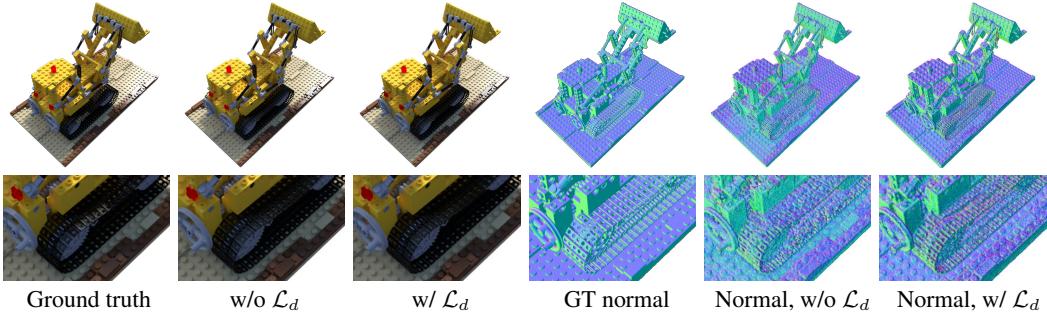


Figure 7: Ablation on SDF continuity regularization \mathcal{L}_d on Lego scene. The estimated normals shown in the figure are SDF computed normals $\hat{\mathbf{n}}$

E ABLATION STUDIES

E.1 SDF REGULARIZATIONS

We show the effects of our purposed SDF regularizations on two synthetic scenes (Lego & Mic) with 80k iterations of training. During the training, we only modify a minimal part of the configurations that are studied and keep the rest of the training configurations unchanged.

Figure 7 first shows how our purposed SDF continuity regularization \mathcal{L}_d (Eqn. 10) improves the quality of the estimated surface normals. Although we cannot observe significant differences in the synthesized images from models with and without the regularization \mathcal{L}_d , we do observe the differences in the estimated normals. The normal computed by the model without the regularization \mathcal{L}_d is more likely to be affected by the texture colors/shadows from the observed training images and is unable to recover the geometry details in those shadowed regions (see the enlarged image patches in Fig. 7). In contrast, the model with our SDF continuity regularization \mathcal{L}_d is unaffected by the coloring or shadowing effects from training images and is able to show more consistent normal directions within a local region.

We also plot the SDF and compositing weight curves along the sampled ray points (as illustrated in Figure 8b) to show more details of the effects of our purposed SDF regularizations. We test all the 4 combinations of our two SDF regularizations \mathcal{L}_s (Eqn. 9) and \mathcal{L}_d (Eqn. 10) in Figure 8. The color differences among 4 computed normal images shown in Figure 8a can be explained by the curves 8c and 8d. When the negative SDF sparsity regularization \mathcal{L}_s is not used (“w/o \mathcal{L}_s ”), the resulting SDF curves have “delayed” zero-level intersection points, i.e., the SDF curves have a long segment of the horizontal line close to the zero-level SDF (the dashed line in 8c) when the sampled ray points approach the ground truth surface. Therefore, the corresponding compositing weight curve (Fig. 8d) is spread across a long range instead of concentrating on a surface point. These flattened compositing weights can severely impact the quality of estimated depths and surfaces. When the SDF continuity regularization \mathcal{L}_d is not used (“w/o \mathcal{L}_d ”), the resulting SDF curves have more oscillations, as shown in Fig. 8c. These oscillating SDF curves can cause the perturbation in the computed normal directions, which also accounts for the noisy normals shown in Figure 7 (“Normal, w/o \mathcal{L}_d ”). In contrast, the model with both \mathcal{L}_s and \mathcal{L}_d (“w/ \mathcal{L}_s , w/ \mathcal{L}_d ”) has the stablest SDF decreasing curve when the sampled ray crosses the object surface. Meanwhile, the corresponding compositing weight curve with one concentrated spike also indicates the more accurate depth and surface estimations.

E.2 NEURAL REPRESENTATION OF THE ENVIRONMENT LIGHT

In this subsection, we evaluate the effect of our neural implicit representation of the environment light by comparing the MLP-based environment light representation and the non-MLP representation. For the non-MLP environment light representation, we directly learn the pixel values on the 32×16 environment light probe image, i.e., each 3-dimensional HDR pixel is treated as the learnable neural parameters, similar to the parameterization in NeRFactor (Zhang et al., 2021b). We keep the rest of the training settings unchanged and run the light decomposition training (Sec. 5.4) for 10k iterations. Figure 9 shows the comparison of the estimated light probe images at different training iterations. We can first observe that the learning of the environment light already saturates after 5k iterations, which means our 10k iterations of training are sufficient for our low-resolution

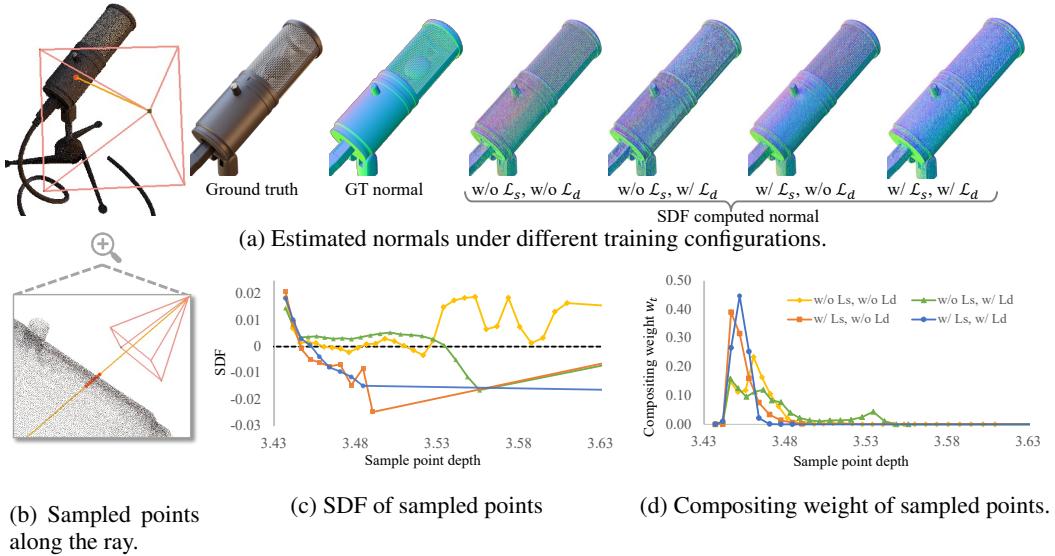


Figure 8: Ablation of our purposed SDF regularizations (\mathcal{L}_s & \mathcal{L}_d) on Mic scene.

environment light estimation. Although non-MLP light representation captures the major lighting patterns in the ground truth light probe images, our coordinate-based MLP representation is able to recover more high frequency details in the estimated environment light (e.g., see the windows in the interior environment used by the Drums scene). Even in the scenes such as Lego that are mainly Lambertian, our MLP representation is still able to infer the high-intensity, concentrated light source (the white spot in our MLP representation for Lego scene⁴) from the observed training images. The high-intensity, concentrated light sources in the light probe image are critical for synthesizing hard cast shadows. As shown in the Figure 10, the oversmoothed light probe image estimated by the non-MLP representation cannot synthesize the shadows (Fig. 10b) for the deformed Lego scene as clear as the results (Fig. 10c) from our MLP-based environment light representation .

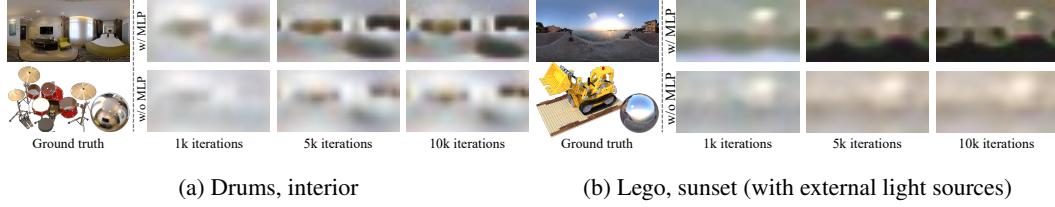


Figure 9: Qualitative comparisons of environment light estimation (“with MLP” vs. “without MLP”) after different training iterations (1k, 5k, 10k).



Figure 10: The illuminations of the deformed Lego scene under different rendering options.

E.3 DECOMPOSITION TRAINING LOSS

We finally evaluate the effect of our decomposed light training loss (Eqn. 12). We use the decomposed training loss in order to learn more accurate BRDF parameters for the specular reflectance.

⁴The Lego scene in NeRF’s Blender synthetic dataset has an external panel light source with the light intensity higher than the light sources in the environment light image.

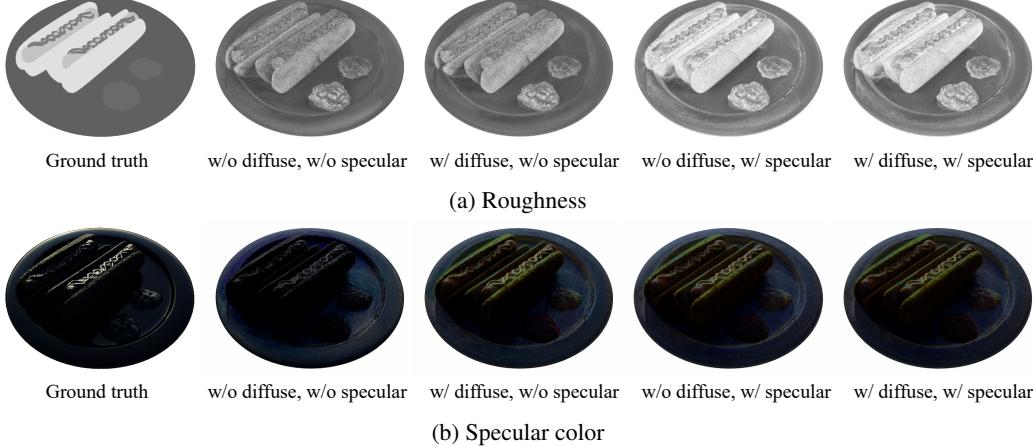


Figure 11: Ablation for the light decomposition training strategy.

Therefore, we compare the estimated roughness and BRDF integrated specular color from varied training strategies for ablation study. We run the lighting decomposition training with 4 combinations of decomposed specular color loss and decomposed diffuse color loss, and show the results of the Hotdog scene in Figure 11. When the models are trained without the decomposed specular color loss (labels containing “w/o specular”), the resulting roughness maps are unable to clearly distinguish the different materials of the plate and the hotdog. The lack of decomposed color losses also leads to much darker synthesized specular colors (“w/o diffuse, w/o specular” in Fig. 11b) Exclusively adding decomposed diffuse color loss can help recover more specular reflectance but has limited improvements in the estimation of the material roughness. In contrast, the decomposed specular color loss can effectively help SPIDR to distinguish the different scene materials. Roughness images from models with specular color loss are more like the ground truth roughness, which indicates a more accurate estimation of the material’s BRDF.

F ADDITIONAL RESULTS

F.1 PER-SCENE BREAKDOWN

Table 2 shows the per-scene breakdown comparison of the quantitative results shown in Table 1. Our SPIDR model has rendering scores comparable to the original Point-NeRF and has significantly better results on the estimated surface normals, especially for shiny scenes or scenes with strong specular highlights (e.g., Materials and Ficus).

F.2 QUALITATIVE RESULTS OF THE RECONSTRUCTED SCENES

In Figure 12 and Figure 13, we present more visual results of the tested scenes from both synthetic and real-captured datasets. In these figures, we visualize our estimated normal, roughness, visibility, and environment light. These are the key factors that control SPIDR’s illumination effects. In addition to the scenes used for quantitative evaluations, we also test our model on two more synthetic scenes (Mannequin and Trex) and two challenging real-captured scenes from BlendedMVS (Evangelion and Gundam). Note that the two real scenes we additionally test have unmasked backgrounds. In order to model the unbounded background, we employ a simplified NeRF++ (Zhang et al., 2020) model (fewer layers, and no view-dependent MLP) to separately represent the background. We use COLMAP (Schönberger et al., 2016) to initialize the point cloud for these two real scenes. The qualitative results show that SPIDR can make convincingly good estimations of surface normals as well as BRDF parameters for most of the scenes. The comparison between the ground truth environment light and our estimated environment light in synthetic scenes also demonstrates the effectiveness of our novel presentation of the environment light.

F.3 ADDITIONAL GEOMETRY EDITING RESULTS

In addition to the editing results shown in the main paper, we showcase more geometry editing results in this part. We choose two BlendedMVS scenes evaluated before (Character and Statues) to perform mesh-guide deformation. To further showcase SPIDR’s capability in rendering accurate illumination in the deformed scenes, we tested two synthetic scenes (Mannequin and Trex) with intentional deformations that can cause light occlusion on the object surface. As shown in Figure

14, SPIDR’s BRDF-based rendering results (the last column) can accurately update the surface illumination caused by occlusions (see the image patches in Fig. 14). Besides, we conduct direct point cloud manipulations on the two unmasked real-captured scenes (Evangelion and Gundam). Results shown in Figure 14 demonstrate that our SPIDR is able to achieve high quality synthesis of the deformed objects in these challenging real captured scenes.

F.4 SCENE RELIGHTING

Although scene relighting is not the main topic of this work, our SPIDR model does support the scene relight, as SPIDR has the complete decomposition of BRDF rendering parameters of the scene. We can simply replace SPIDR’s estimated environment light with the target HDR environment light image (HDRI) to achieve the scene relighting. We show scene relighting results (some are also combined with scene deformations) in Figure 15. These relighting results also show one limitation of SPIDR, that is, SPIDR cannot completely remove the shadows existing in the observed training images. However, complete shadow removal from images under the same environment illumination is an extremely difficult task, since shadows can be misinterpreted as the object’s texture color. Future work on learning decomposed BRDF parameters from images under multiple environments would help SPIDR achieve better results on relighting.

Table 2: Per-scene Quantitative Results

	NeRF Synthetic									Blended MVS			
	Chair	Drums	Lego	Mic	Materials	Ship	Hotdog	Ficus	Jade	Fountain	Character	Statues	
PSNR ↑													
NSVF	33.00	25.18	32.54	34.27	32.68	27.93	37.14	31.23	26.96	27.73	27.95	24.97	
VolSDF	30.57	20.43	29.46	30.53	29.13	25.51	35.11	22.91	25.20	24.05	25.59	24.26	
PointNeRF	35.40	26.06	35.04	35.95	29.61	30.97	37.30	36.13	26.14	25.68	29.06	25.97	
Ours	34.84	25.70	34.77	34.47	27.42	30.74	36.82	33.66	25.97	27.25	27.80	25.50	
Ours-BRDF	30.33	23.21	29.67	27.88	23.74	26.52	32.72	28.21	25.00	24.44	25.42	23.95	
SSIM ↑													
NSVF	0.968	0.931	0.960	0.987	0.973	0.854	0.980	0.973	0.901	0.913	0.921	0.858	
VolSDF	0.949	0.893	0.951	0.969	0.954	0.842	0.972	0.929	0.919	0.908	0.939	0.895	
PointNeRF	0.991	0.954	0.988	0.994	0.971	0.942	0.991	0.993	0.931	0.935	0.970	0.930	
Ours	0.991	0.957	0.989	0.993	0.952	0.945	0.991	0.990	0.935	0.957	0.968	0.933	
Ours-BRDF	0.974	0.929	0.969	0.976	0.909	0.897	0.979	0.970	0.898	0.922	0.950	0.910	
LPIPS ↓													
NSVF	0.043	0.069	0.029	0.010	0.021	0.162	0.025	0.017	0.094	0.113	0.074	0.171	
VolSDF	0.056	0.119	0.054	0.191	0.048	0.191	0.043	0.068	0.128	0.177	0.091	0.196	
PointNeRF	0.010	0.055	0.011	0.007	0.041	0.070	0.016	0.009	0.091	0.104	0.033	0.082	
Ours	0.010	0.049	0.010	0.008	0.046	0.079	0.014	0.015	0.079	0.061	0.039	0.087	
Ours-BRDF	0.037	0.085	0.029	0.037	0.089	0.127	0.043	0.040	0.126	0.098	0.055	0.115	
MAE° ↓													
VolSDF	14.09	21.46	26.62	19.58	8.28	16.97	12.17	39.80	32.17	24.87	28.74	18.03	
PointNeRF	37.24	54.13	40.97	47.51	60.41	50.82	32.61	61.01	50.90	44.59	58.30	54.61	
Ours-Grad	26.58	47.93	25.46	26.15	27.90	27.72	20.56	39.92	50.52	50.07	50.43	38.89	
Ours-Pred	22.19	39.86	24.05	22.06	22.26	23.03	18.23	41.95	43.32	34.42	33.96	27.42	

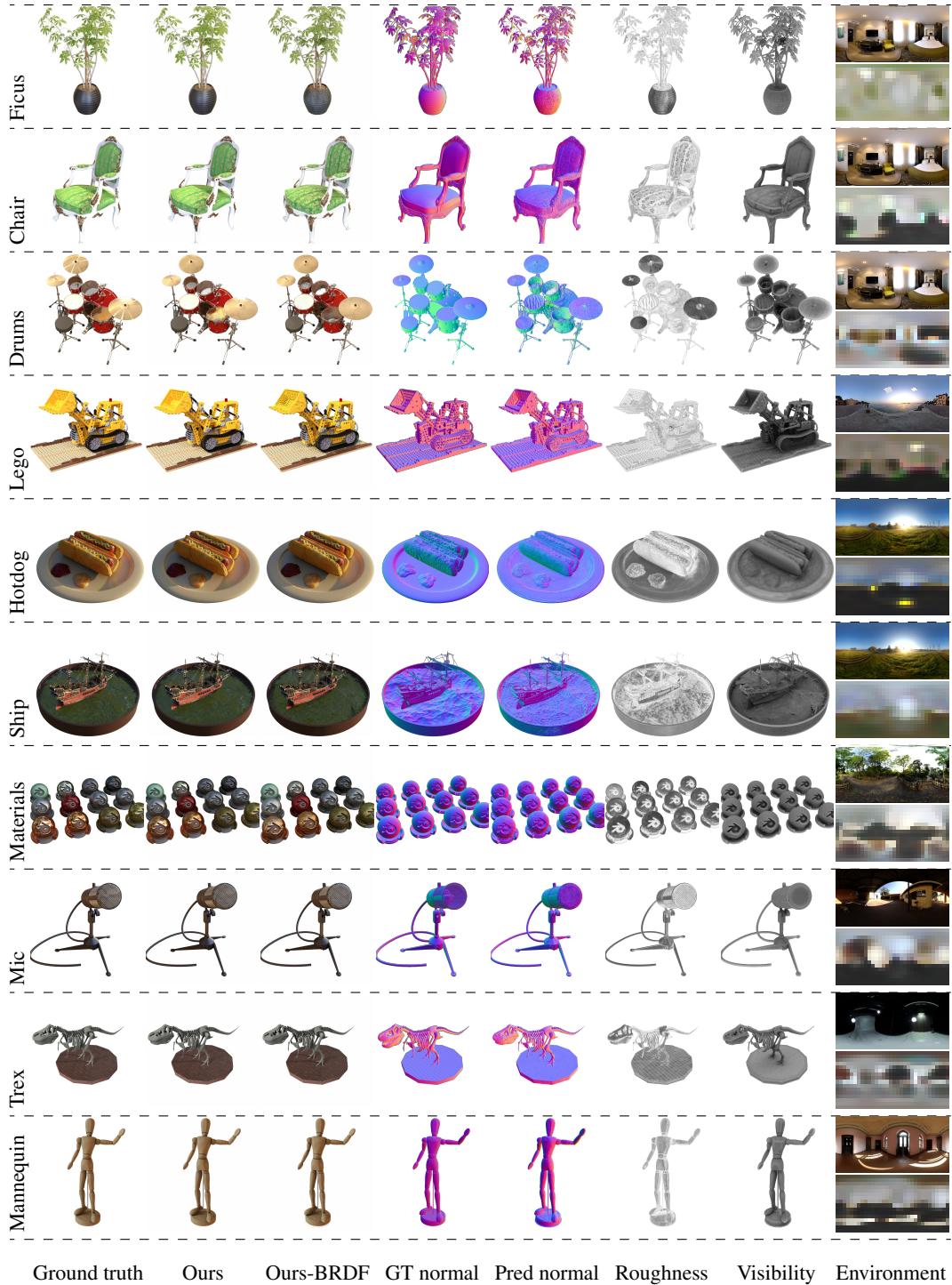


Figure 12: Qualitative results of the synthetic scenes. Note that in the “Environment” column, the upper row shows the ground truth environment light, and the lower row shows our estimated environment light.

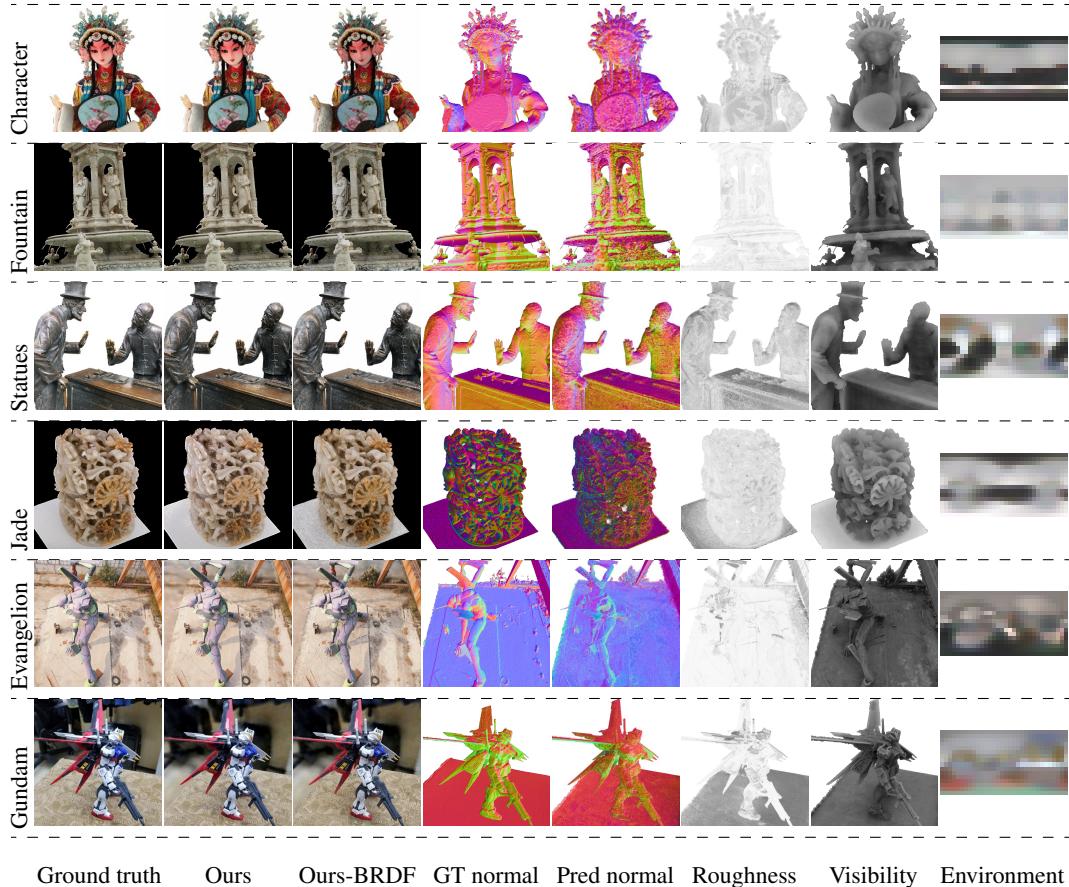


Figure 13: Qualitative results of real-captured scenes from BlendedMVS dataset. Since these real-captured scenes do not have ground truth environment light, we only show our estimated environment light in the last column.

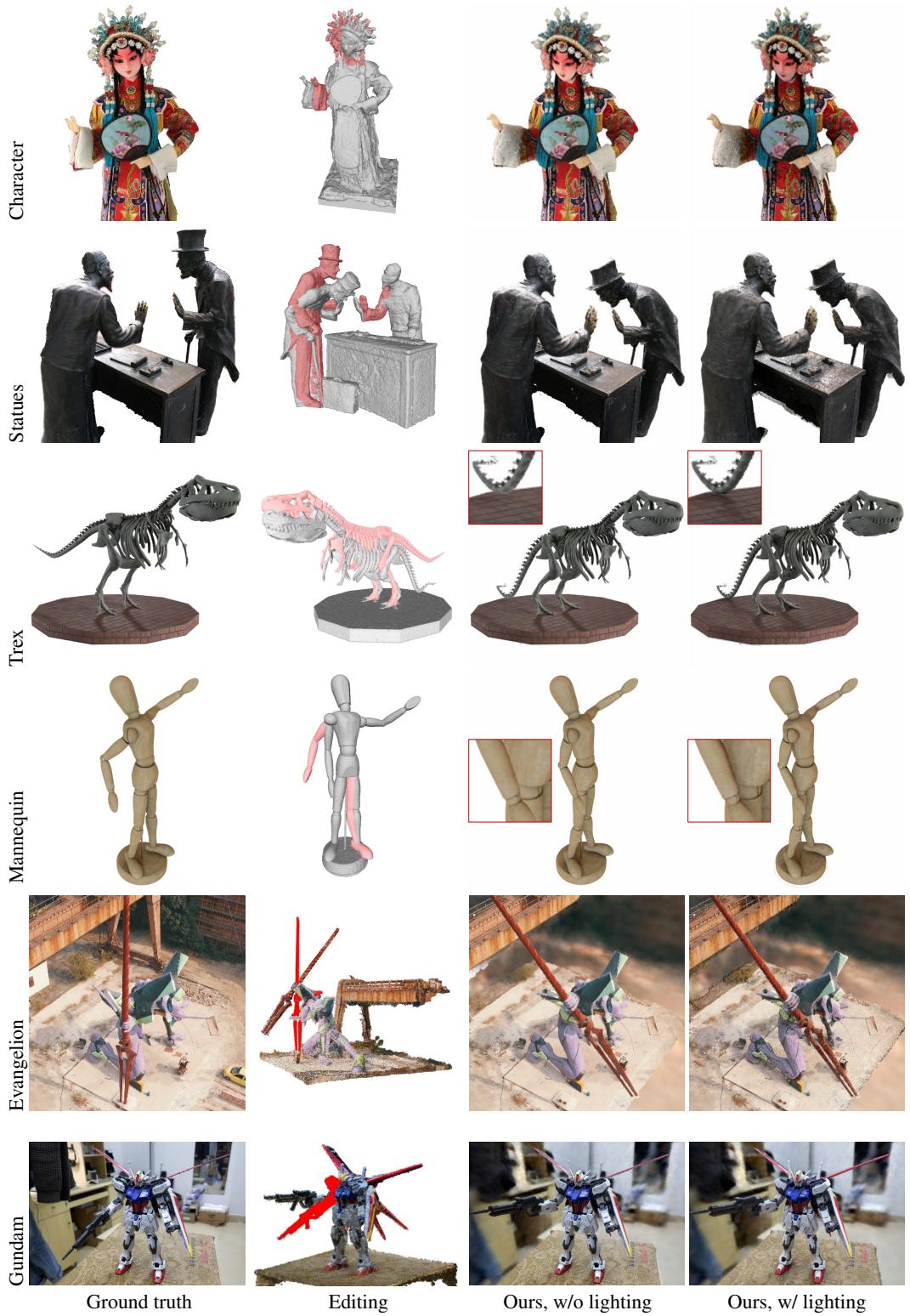


Figure 14: Qualitative results on geometry editing.

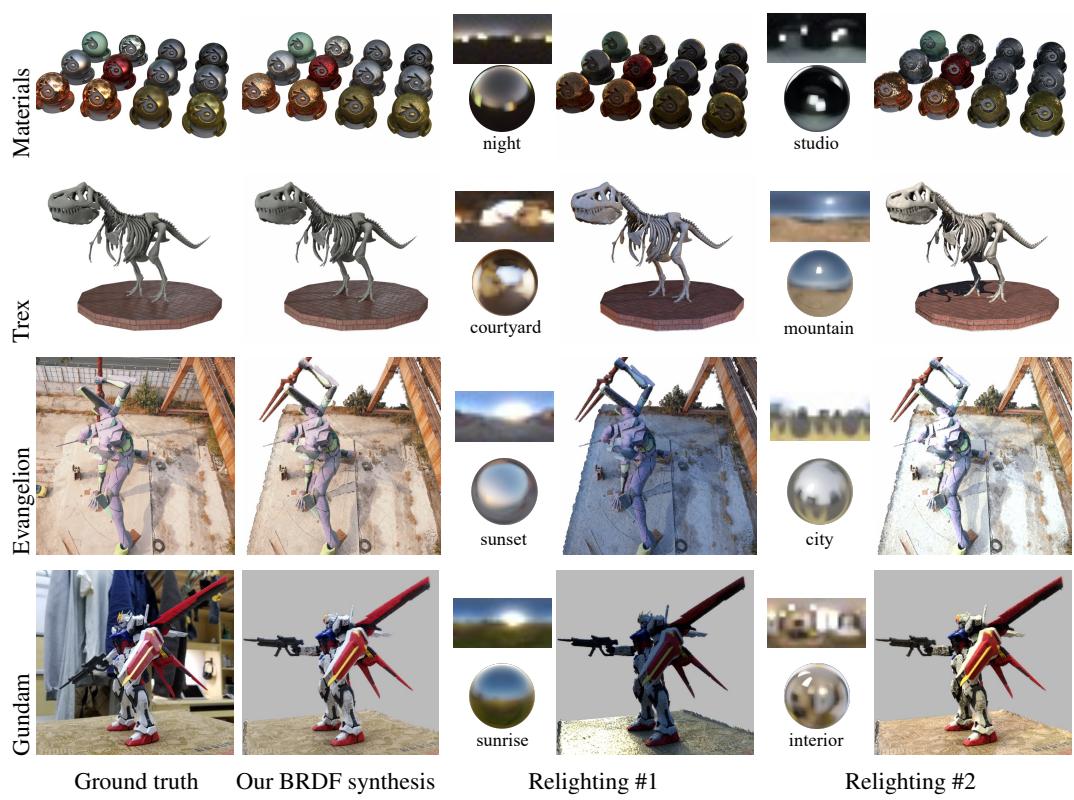


Figure 15: Scene relighting results.