DE-NeRF: DEcoupled Neural Radiance Fields for View-Consistent Appearance Editing and High-Frequency Environmental Relighting

Tong Wu
Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
China
wutong19s@ict.ac.cn

Jia-Mu Sun
Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
China
sunjiamu21s@ict.ac.cn

Yu-Kun Lai
School of Computer Science and Informatics, Cardiff University
United Kingdom
LaiY4@cardiff.ac.uk

Lin Gao*
Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
China
gaolin@ict.ac.cn

ABSTRACT

Neural Radiance Fields (NeRF) have shown promising results in novel view synthesis. While achieving state-of-the-art rendering results, NeRF usually encodes all properties related to geometry and appearance of the scene together into several MLP (Multi-Layer Perceptron) networks, which hinders downstream manipulation of geometry, appearance and illumination. Recently researchers made attempts to edit geometry, appearance and lighting for NeRF. However, they fail to render view-consistent results after editing the appearance of the input scene. Moreover, high-frequency environmental relighting is also beyond their capability as lighting is modeled as Spherical Gaussian (SG) and Spherical Harmonic (SH) functions or a low-resolution environment map. To solve the above problems, we propose DE-NeRF to decouple view-independent appearance and view-dependent appearance in the scene with a hybrid lighting representation. Specifically, we first train a signed distance function to reconstruct an explicit mesh for the input scene. Then a decoupled NeRF learns to attach view-independent appearance to the reconstructed mesh by defining learnable disentangled features representing geometry and view-independent appearance on its vertices. For lighting, we approximate it with an explicit learnable environment map and an implicit lighting network to support both low-frequency and high-frequency relighting. By modifying the view-independent appearance, rendered results are consistent across different viewpoints. Our method also supports high-frequency environmental relighting by replacing the explicit environment map with a novel one and fitting the implicit lighting network to the novel environment map. Experiments show that our method achieves better editing and relighting performance both quantitatively and qualitatively compared to previous methods.

*Corresponding author is Lin Gao (gaolin@ict.ac.cn).

Figure 1: Given a set of input images, we train a neural radiance field that decouples geometry, appearance, and lighting. Our method supports not only the geometry manipulation and appearance editing but also the rendering of the captured or modified scene in a novel lighting condition.

CCS CONCEPTS

• **Computing methodologies** → *Image-based rendering*.

KEYWORDS

neural radiance fields, inverse rendering, editing

1 INTRODUCTION

Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] have shown promising results in scene reconstruction and novel view synthesis. Compared with traditional geometry and appearance representations, such as textured meshes, NeRF does not require precise geometry and texture reconstruction and can produce realistic rendering results. However, besides visualization, editing is also an important task in computer graphics. Traditional 3D modeling applications allow users to edit mesh geometry via modifying face connections or vertex locations and edit the appearance by painting from a given viewpoint. Lighting conditions are also changeable by replacing the environment map. But in conventional NeRF, the geometry is represented by a density function that does not well reflect the real geometry and its appearance is an entanglement of material and lighting, which increases the difficulty of editing.

On the geometry editing side, a few methods propose to deform neural radiance fields by deforming sample points on a ray [Garbin et al. 2022; Peng et al. 2022; Xu and Harada 2022; Yuan et al. 2022]. For appearance editing, researchers try to decompose geometry, material and lighting from 2D images in an implicit way so that each component can be edited independently. PhySG [Zhang et al. 2021a] and NeRD [Boss et al. 2021a] use MLP (Multi-Layer Perceptron) networks to predict BRDF (Bidirectional Reflectance Distribution Function) materials and approximate lighting with Spherical Gaussian functions. But their geometry is still in an implicit form and lighting representation is smooth so high-frequency environmental relighting is beyond their limits. For better material estimation, NeRFactor [Zhang et al. 2021b] predicts material parameters with a pre-trained BRDF decoder and represents lighting with a low-resolution image, which prevents it from representing high-frequency lighting. RefNeRF [Verbin et al. 2022] proposes not to explicitly decompose BRDF materials but instead learns view-dependent and view-independent appearance simultaneously. Although achieving high-quality reconstruction results, RefNeRF [Verbin et al. 2022] can only edit a scene by adjusting its color network's outputs and is unable to deform the geometry or relight the input scene.

To transfer editing from one viewpoint to other viewpoints seamlessly, NeuTex [Xiang et al. 2021] maps sample points to a unified 2D texture space and uses traditional UV mapping to query corresponding colors. After training, the appearance of the scene is baked into the 2D texture image. Users can edit the neural radiance field by painting the 2D texture image. However, the 2D texture generated by NeuTex [Xiang et al. 2021] is usually distorted and hard to be edited. To resolve this issue, NeuMesh [Bao et al. 2022] defines learnable geometry and appearance features on a pre-reconstructed mesh for the scene and learns to decompose the geometry and appearance using two MLP networks. Unfortunately, its appearance is still an entanglement of material and lighting so that rendered results can be inconsistent with the input editing when viewed from novel viewpoints and its lighting conditions cannot be changed.

To allow view-consistent appearance editing and high-frequency environmental relighting, we propose DE-NeRF that decouples the geometry, appearance and lighting of the input scene. Given a set of captured 2D images for a scene, we first reconstruct its geometry with an SDF (Signed Distance Field) network. Then we define the geometry and view-independent appearance features on the reconstructed mesh's vertices and use the corresponding geometry network and appearance network to predict signed distance values and appearance parameters. By baking geometry and view-independent appearance features onto mesh vertices, DE-NeRF can seamlessly transfer the appearance editing from one viewpoint to other viewpoints and the edited appearance is consistent across different viewpoints. For lighting, we propose to use a hybrid representation, composed of an explicit low-resolution environment map for efficiency and an implicit lighting network. The explicit environment map is responsible for low-frequency diffuse lighting and the implicit lighting network is trained to represent specular lighting. After training, geometry, view-independent appearance and lighting are disentangled and they can be separately edited without influencing other components.

Our contributions can be summarized as follows:

- A neural radiance fields editing method that allows editing of geometry, appearance and lighting. Appearance editing from one viewpoint can be seamlessly transferred to other viewpoints and the rendered results are view-consistent after editing.
- Our lighting representation supports high-frequency environmental relighting and produces more faithful relighting results compared to previous methods.

2 RELATED WORK

2.1 Neural Geometry Reconstruction

With the development of neural rendering [Oechsle et al. 2019; Thies et al. 2019] and implicit geometry representations [Chen and Zhang 2019; Mescheder et al. 2019; Park et al. 2019], *surface-based* rendering methods [Niemeyer et al. 2020; Yariv et al. 2020; Zhang et al. 2021c] are proposed to learn geometry and appearance separately to reconstruct an object's geometry from 2D images by minimizing the difference between rendered images and input images. Later with the emergence of Neural Radiance Fields (NeRF) [Mildenhall et al. 2020], researchers start to work on geometry reconstruction with *volume rendering*. A pioneering work that builds the connection between implicit geometry representations and neural radiance fields is NeuS [Wang et al. 2021], which derives an unbiased and occlusion-aware formulation for the neural radiance field's density function from a signed distance function (SDF). UNISURF [Oechsle et al. 2021] instead treats geometry as an occupancy field that predicts whether a sampled point is on the object surface and replaces the alpha value in volume rendering with the occupancy value. Yariv et al. [2021] also transform the SDF to a density function in volume rendering and their transformation function is the Cumulative Distribution Function (CDF) of a learnable Laplace distribution. To reduce the requirement for the number of input images, SparseNeuS [Long et al. 2022] extracts 2D features from images to provide extra information for sample points in the space via projection. To accelerate the training process of geometry reconstruction, VOXURF [Wu et al. 2022] defines learnable features on voxel grids similar to [Fridovich-Keil et al. 2022; Liu et al. 2020] to speed up training.

2.2 NeRF Decomposition

Recently, researchers started to disentangle geometry, material and lighting from Neural Radiance Fields. NeRV [Srinivasan et al. 2021] decomposes BRDF materials under a given lighting condition. It models direct illumination and one-bounce indirect illumination and uses a network to predict the visibility of the sample point. NeRD [Boss et al. 2021a] approximates lighting with Spherical Gaussian (SG) functions and reduces the learning difficulty by first extracting view-independent material parameters and density functions and applying them to the learning of view-dependent material parameters. For more accurate material estimation, Boss et al. [2021b] predict BRDF materials with a material autoencoder pre-trained on a BRDF material dataset [Matusik et al. 2003]. NeROIC [Kuang et al. 2022] approximates lighting with Spherical Harmonic (SH) coefficients and decomposes static appearance and transient appearance. NeRFactor [Zhang et al. 2021b] is the first work to learn shadow decomposition under unknown lighting conditions. Similar to [Boss et al. 2021a], it first trains a standard NeRF network to determine the geometry. Then it predicts material with a pre-trained BRDF decoder and optimizes its lighting which is represented by a low-resolution image. More recently, RefNeRF [Verbin et al. 2022] implicitly decomposes view-dependent appearance and view-independent appearance via two separate networks and can learn high-frequency specular reflections, but it does not decompose shadow or lighting. Besides the works mentioned above, there are works that decompose scenes based on other representations. PhySG [Zhang et al. 2021a] models geometry as an SDF network and its lighting is approximated by a composition of several Spherical Gaussian (SG) functions [Wang et al. 2009]. It utilizes the Disney BRDF model [Bi et al. 2020] and assumes that the scene can only have one single specular BRDF material, causing a performance drop on more complex scenes. InvRender [Zhang et al. 2022b] further models indirect illumination with another set of SG functions to handle more complicated appearances like inter-reflection. NvdiffRec [Munkberg et al. 2022] and NvdiffRecMC [Hasselgren et al. 2022] use Deep Marching Tetrahedra [Shen et al. 2021] as its geometry representation and learn to decompose the input scene with differential rasterization rendering [Laine et al. 2020] and differentiable Monte Carlo renderer. They handle high-frequency lighting but struggle when the objects have a highly glossy surface.

2.3 Neural Radiance Field Editing

Classified by editing targets, previous works can be roughly divided into geometry editing and appearance editing. In terms of geometry, several works [Garbin et al. 2022; Xu and Harada 2022; Yuan et al. 2022] share a similar idea to reconstruct an explicit mesh as a proxy for a static scene and builds correspondence between the mesh and NeRF. By editing the mesh using As-Rigid-As-Possible deformation [Sorkine-Hornung and Alexa 2007], sample points in the rendering process are transformed along with the mesh via barycentric coordinate interpolation. For appearance editing, several methods [Huang et al. 2022; Wang et al. 2022; Zhang et al. 2022a] propose to edit the appearance of NeRF by stylizing it with an image or text prompt. EditNeRF [Liu et al. 2021] is the first work that allows users to edit NeRF by editing 2D images, which greatly reduces the editing difficulty. It models a scene with a shape code and a color code. Editing is performed by optimizing the color code. But it requires a large dataset from the same category to generate plausible editing results. NeuTex [Xiang et al. 2021] maps sample points in a single scene to UV coordinates and gets its color from a learnable UV map. After training, the appearance of NeRF can be edited by painting the UV texture. However, the learned UV mapping is usually distorted and hard to be edited. NeuMesh [Bao et al. 2022] reconstructs the geometry of the scene using NeuS [Wang et al. 2021] and defines learnable geometry and appearance features on mesh vertices. It allows users to edit NeRF's appearance from 2D images by optimizing appearance features similar to EditNeRF [Liu et al. 2021]. Since its appearance features do not disentangle material and lighting, artifacts may occur when observed from a different viewpoint after editing. Our method focuses on decoupling NeRF into geometry, appearance and lighting for independent editing, where the geometry and view-independent appearance are encoded on mesh vertices to ensure view consistency, and a hybrid lighting representation is proposed to support relighting with high-frequency environmental lighting.

3 METHOD

We propose DE-NeRF, a decoupled geometry, appearance and lighting editing method for NeRF that allows view-consistent appearance editing and high-frequency environmental relighting. The pipeline of our method is illustrated in Fig. 2. We first reconstruct the geometry of the input scene (Sec. 3.1). To enable geometry and appearance editing, we define learnable features for geometry and appearance on the vertices of the reconstructed mesh to bake view-independent information onto the reconstructed mesh to ensure view consistency. For lighting, we propose a hybrid lighting representation that supports both low-frequency lighting and high-frequency lighting. The low-frequency lighting is modeled by an explicit environment map where each pixel in it represents a light and all lights in the environment map are integrated at every sample point in the scene. For high-frequency lighting, it is costly to represent it with a large environment map. Instead, we model it with an implicit lighting network and encourage it to be consistent with the explicit environment map. Under the guidance of the reconstructed geometry and the input images, we decouple the geometry, appearance and lighting of the scene by optimizing the learnable features on the mesh vertices, the learnable environment map, and the lighting network (Sec. 3.2). After decoupling, users can edit the geometry, appearance, and lighting of the input scene (Sec. 3.3).

3.1 Geometry Reconstruction

Recent neural implicit representations [Chen and Zhang 2019; Mescheder et al. 2019; Park et al. 2019] and neural rendering techniques [Mildenhall et al. 2020] have achieved great success in the scene reconstruction task. In this work, we use the Signed Distance Function (SDF) as our geometry representation for smooth geometry reconstruction. The SDF can be parameterized as an MLP network $s = F(x)$. It takes a sample point $x(t) = o+v\cdot t$ as input and outputs its signed distance s to the surface, where o is the origin of a camera ray, v is the ray direction, and t is the parameter that

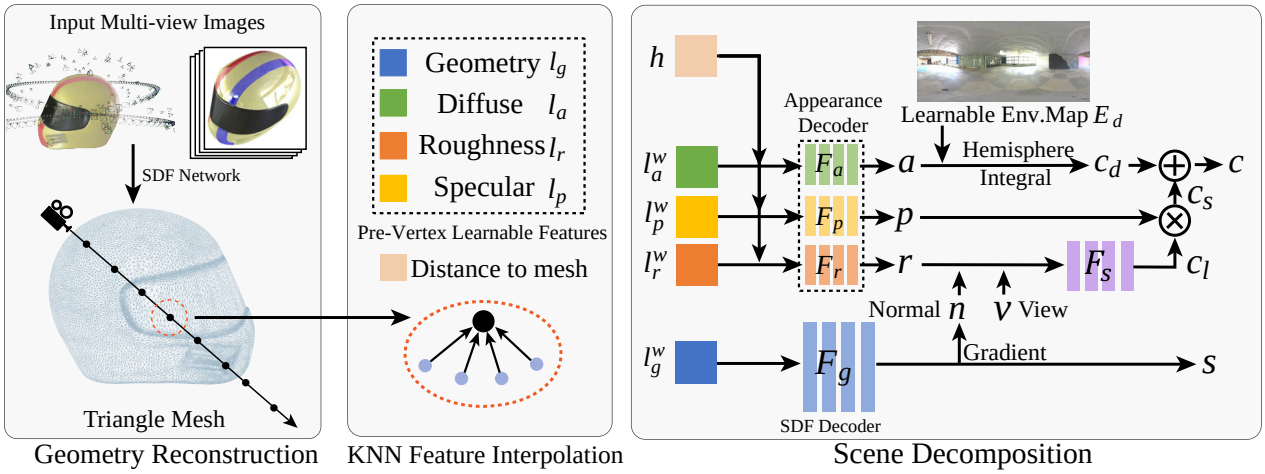


Figure 2: Given a set of images, we learn a signed distance function to reconstruct the geometry. Then, on the vertices of the reconstructed mesh, we set up learnable geometry features l_g and appearance features l_a, l_r, l_p (corresponding to diffuse, roughness and specular components) to decompose geometry, appearance, and lighting in the scene. A sample point's geometry feature l_g^w and appearance features l_a^w, l_r^w, l_p^w are obtained by KNN (K-nearest neighbor) interpolation. The geometry feature l_g^w and the distance to the mesh h are fed into an SDF decoder to predict its signed distance value s. Similarly, appearance features l_a^w, l_r^w, l_p^w, and distance h go through several appearance decoders to predict diffuse albedo a, roughness value r, and specular tint p. A learnable environment map E_d is integrated with the diffuse albedo to get diffuse color c_d. We also train a specular lighting decoder F_s to predict specular lighting c_l, which is multiplied by the specular tint t to produce the specular color c_s. Combining c_d and c_s, we get the color c for this point.

determines the sample point on the ray. To learn the SDF from multi-view images of the scene, we adopt the occlusion-aware and unbiased volume rendering technique from NeuS [Wang et al. 2021] to render the SDF of the scene. Same as NeuS, we define the geometry density based on SDF as $\sigma(t) = \max\left(-\frac{\frac{d\Phi_s}{dt}(f(x(t)))}{\Phi_s(f(x(t)))}, 0\right)$, where $\Phi_s(x) = (1+e^{-sx})^{-1}$ and s is a trainable deviation parameter.

Generally, this formulation works well. However, for scenes with specular reflection, a point on the surface can present totally different colors when observed from different viewpoints, making it hard to be learned by a single color network conditioned on the viewpoint as NeuS does. To fake the complicated view-dependent effects, NeuS tends to wrongly construct a concave surface so that, from different viewpoints, the camera will not see the same surface point but different points with different colors. To address this issue, we divide the color network into two branches following RefNeRF [Verbin et al. 2022] to model view-independent appearance and view-dependent appearance respectively, which reduces the learning difficulty of the color network. The view-independent branch takes a sample point as input and outputs its view-independent color c_d and its specular tint p. Both the sample point and the ray direction v are fed into the view-dependent branch to predict the view-dependent color c_l. The final color of a sample point can be formulated as $c = c_d + p \cdot c_l$.

To calculate the color of each camera ray $C(v)$, we integrate the colors of the sample points on the ray by the volume rendering equation: $C(v) = \sum_{i=1}^{N} T_i \alpha_i c_i$, where T_i is accumulated transmittance defined as $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$; and α_i represents opaque value at point x_i. We learn to reconstruct the input scene's geometry and appearance by optimizing the following loss function:

$$L_g = L_c + \lambda L_e = \sum_{v \in V} \left\| C(v) - C^t(v) \right\| + \lambda \sum_{v \in V} \sum_{i=1}^{N} \left\| \|\nabla_{x_{v,i}}\| - 1 \right\|_2^2, \quad (1)$$

where V the camera rays in a training batch. $C^t(v)$ represents the ground truth pixel color for a ray v. $x_{v,i}$ is the ith sample point on the ray v. $\|\nabla_{x_{v,i}}\|$ is the spatial norm of the SDF network $F(x)$'s gradient at point $x_{v,i}$.

3.2 Scene Decoupling

After reconstructing the scene's geometry and appearance, we extract a mesh using the marching cubes [Lorensen and Cline 1987] algorithm. To decouple geometry, appearance and lighting components for editing, we define learnable features on the vertices of the mesh, denoted as l_g for geometry features, l_a for diffuse features, l_p for specular features, and l_r for roughness features. For a sample point x, its features $l_g(x), l_a(x), l_p(x), l_r(x)$ are defined by the weighted average of its K nearest neighbors from the reconstructed mesh vertices as $l_*^w(x) = \frac{\sum_{i=0}^{K} w_i(x) l_{*,i}(x)}{\sum_{i=0}^{K} w_i(x)}$ similar to NeuMesh [Bao et al. 2022] and PointNeRF [Xu et al. 2022]. $l_*^w(x)$ represents the interpolated learnable features, i.e., $l_g^w(x), l_a^w(x), l_p^w(x)$ and $l_r(x)$. the weight $w_i(x)$ is the inverse of the distance between x and its ith nearest neighbor x_i:

$$w_i(x) = \frac{1}{\|x_i - x\|_2}. \quad (2)$$

Next, we use a geometry network that takes the geometry feature $l_g^w(x)$ and the distance $h(x)$ from x to the reconstructed mesh as input to predict the signed distance value s of point x. The distance

$h(x)$ is also calculated by the weighted average of the distances to its K nearest neighbors, where the weights are defined in Eqn. 2. Similarly, we feed the features l_a^w, l_p^w, l_r^w into separate MLPs to infer diffuse albedo a, specular tint p, and roughness value r. The signed distance value and appearance parameter predictions can be formulated as follows:

$$s = F_g(l_g^w, h); a = F_a(l_a^w, h); p = F_p(l_p^w, h); r = F_r(l_r^w, h). \quad (3)$$

On the lighting side, the diffuse lighting is represented by an explicit environment map E_d where each pixel can be seen as a light so that the diffuse color c_d for a point can be obtained by integrating all lights in the environment map E_d at this point via $c_d = \int_\Omega \frac{a}{\pi} L_i n \cdot \omega_i d\omega_i$. ω_i is the direction of incident light L_i. n is the normal direction for point x derived by the gradient of the geometry network F_g and $\cdot$ denotes dot product.

For specular lighting that may contain high-frequency details, it is costly to represent it with a high-resolution environment map and integrate the environment map and the material parameters using the rendering equation. Inspired by the Split-Sum [Karis 2013] approximation in real-time rendering and the recent work RefNeRF [Verbin et al. 2022] that decouples lighting from the rendering equation, we model a sample point's specular color $c_s = p \cdot c_l$ as the multiplication of its specular tint p and the light color c_l that comes from the reflected direction $\omega_r = 2(\omega_o \cdot n) - \omega_o$ of the view direction $\omega_o = -v$ *w.r.t.* its normal direction n. Here, the light color c_l is predicted by a specular lighting decoder $F_s(\cdot)$ that takes a sample point's roughness r, the dot product $\cos\theta = n \cdot \omega_o$ of the normal direction n and the view direction ω_o, and the reflected direction ω_r as input:

$$c_l = F_s(r, \cos\theta, \omega_r). \quad (4)$$

Combining the diffuse color c_d and the specular color c_s, we get the sample point's color $c = c_d + c_s$ and render a pixel color using volume rendering. For training, we minimize the following loss:

$$L = L_c + L_{sdf} + \lambda_1 L_e + \lambda_2 L_{gs} + \lambda_3 L_{ec}, \quad (5)$$

where L_c and L_e are the same as those in Eqn. 1. L_{sdf} is the loss between the predicted signed distance value s at a sample point and the ground truth signed distance value s^t to the reconstructed mesh.

$$L_{sdf} = \sum_{v \in V} \sum_{i=1}^{N} \left\| s_{v,i} - s_{v,i}^t \right\|_2^2. \quad (6)$$

L_{gs} is a smoothness loss that penalizes differences between adjacent vertices' geometry features and is defined as:

$$L_{gs} = \sum_i \sum_{j \in \mathcal{N}(i)} \|l_{g_i} - l_{g_j}\|_2, \quad (7)$$

where $\mathcal{N}(i)$ is the indices of the adjacent vertices for the ith vertex.

L_{ec} denotes the environment map consistency loss, which enforces the environment map E_d to be consistent with the specular lighting generated by the specular lighting decoder F_s. However, our specular lighting has an implicit representation so it is impossible to directly compare it with diffuse lighting. Recall Eqn. 4 that the light color c_l from the reflected direction ω_r at a point with roughness r is $F_s(r, \cos\theta, \omega_r)$. Following the approximation in SplitSum [Karis 2013], when the reflected direction ω_r is the same as the view direction ω_o, the normal direction is the same as the reflected direction ω_r and the view direction ω_o, so $\cos\theta = \cos 0 = 1$. In this case, the output of the specular lighting decoder F_s is an approximation of the environment map as shown in Fig. 3. Thus L_{ec} is defined as:

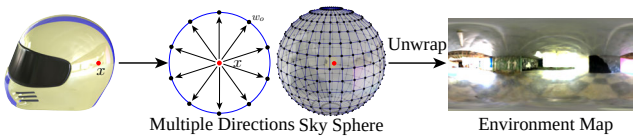


Figure 3: Given a sample point in the scene (the red point), we sample multiple directions ω_o from the sample point to points (black points on the blue frame) on the sky sphere. We treat these directions as view directions and feed them along with the roughness value of the sample point into the specular lighting decoder to get the specular lighting colors from different view directions. These predicted specular lighting colors are unwrapped to the 2D image space as an environment map.

$$L_{ec} = \sum_{j=1}^{P} ||F_s(r, 1, \omega_{o_j}) - E_d(\omega_{o_j})||, \quad (8)$$

where P is the number of pixels in the environment map E_d. r is the roughness value for a randomly sampled point on the mesh surface and ω_{o_j} is the jth unit vector starting from the origin to the jth pixel's location in E_d on an extremely large sky sphere.

3.3 Scene Editing

With geometry, appearance and lighting decoupled by the network, our method allows users to edit each component individually without affecting other components. For example, lighting can be changed without influencing the geometry or appearance. At a finer level, we can also edit appearance parameters like diffuse albedo, roughness, and specular tint independently. In the following, we elaborate on how to edit each component.

3.3.1 Geometry Editing. Similar to NeuMesh [Bao et al. 2022], we apply As-Rigid-As-Possible deformation [Sorkine-Hornung and Alexa 2007] to the reconstructed mesh to deform the scene.

3.3.2 Appearance Editing. Our appearance editing supports editing all appearance features, including diffuse, specular and roughness components by painting a rendered image of the scene. Given a painted image, we can locate the corresponding mesh vertices for editing by applying raycasting from the camera to the reconstructed mesh. The appearance features l_*^e of these vertices are then treated as trainable parameters while the features of other vertices remain the same. The optimization target function can be formulated as follows:

$$\arg\min_{l_*^e} \sum_{v \in V^e} ||C_\#(v) - C_e(v)||, * \in \{a, r, p\}, \quad (9)$$

where V^e denotes the corresponding camera rays of the painted pixels. $C_e(v)$ stands for the color of a painted pixel. $C_\#(v)$ is a rendered component's pixel color after volume rendering, e.g., the diffuse color c_d.

3.3.3 Relighting. As mentioned in Sec. 3.2, our lighting mechanism has two parts, namely diffuse lighting and specular lighting. The diffuse lighting is represented by an explicit environment map and the specular lighting is represented by an MLP network. For relighting, the diffuse lighting can be easily changed by replacing the environment map with the target environment map. However, as the specular lighting has an implicit representation, it cannot be directly changed. Instead, we optimize the specular lighting network F_s to fit the target environment map E^t by minimizing the following loss:

$$L_{relight} = \sum_{i=1}^{S}\sum_{j=1}^{P} ||F_s(r_i, 1, \omega_{o_j}) - E^t(\omega_{o_j})||, \tag{10}$$

where S denotes the number of sample points on the mesh surface, and P is the number of pixels in the target environment map image E^t. r_i is the roughness value of the ith sample point, respectively, and ω_{o_j} is the jth unit vector starting from the origin to the jth light's location in the target environment map E^t on an extremely large sphere. Note that we make the same assumption as in Eqn. 8 that the normal direction n is the same as the view direction ω_o, so $\cos\theta = \cos 0 = 1$.

However, Eqn. 10 only works for those sample points with small roughness values so that it can well preserve the lighting from the environment map. Directly applying Eqn. 10 to those sample points with large roughness values may result in unexpected results, such as a rough surface looking like a mirror after relighting (please refer to Fig. 7). Thus, we construct a mipmap of the target environment map by computing pre-filtered environment maps at different roughness levels by Monte-Carlo sampling:

$$L(j) = \int_{\Omega} L_i(\omega_i)\,(\omega_i \cdot n)\,d\omega_i \approx \frac{\sum_{j=1}^{J} L_i(\omega_i^j)\left(\omega_i^j \cdot n\right)}{\sum_{j=1}^{J}\left(\omega_i^j \cdot n\right)} \tag{11}$$

where $L_i(\omega_i)$ is the light coming from direction ω_i and J is the number of sampled incident light directions. The sampling process is determined by the roughness value and can be quickly performed using [Krivánek and Colbert 2008]. After integrating over incoming lighting at different roughness levels, we can construct a mipmap of the environment map which has a fixed roughness value at each mip level. The specular lighting can be quickly queried from the mipmap based on the sample points' roughness r and the view direction ω_o. So Eqn. 10 can be further improved:

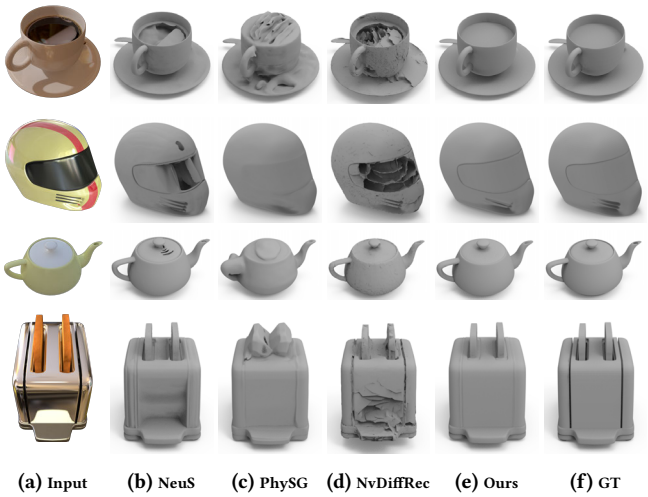
$$L_{relight} = \sum_{i=1}^{S}\sum_{j=1}^{P} ||F_s(r_i, 1, \omega_{o_j}) - M(\omega_{o_j}, r_j)|| \tag{12}$$

where M is the pre-filtered environment mipmap computed by Eqn. 11 and $M(\omega_{o_j}, r_j)$ is the light color viewed from direction ω_{o_j} and interpolated by roughness r_j.

4 RESULTS AND EVALUATIONS

4.1 Datasets and Evaluation metrics

We conduct our experiments on two synthetic datasets, NeRF Synthetic [Mildenhall et al. 2020] and Shiny Blender [Verbin et al. 2022] datasets, and the real DTU [Jensen et al. 2014] dataset. To evaluate the quality of the reconstructed meshes, we use Chamfer Distance between the reconstructed meshes and the corresponding ground truth geometry. Regarding rendering quality, we use SSIM [Wang et al. 2004], PSNR, and LPIPS [Zhang et al. 2018] metrics to evaluate the similarity between the rendered images and the corresponding ground truth images. For editing results, we evaluate the image quality by calculating the Fréchet Inception Distance (FID) [Heusel et al. 2017] between the image set before editing and after editing, which has been widely used in image generation and editing tasks. For training details and the network architecture, please refer to the supplementary material.



(a) Input **(b) NeuS** **(c) PhySG** **(d) NvDiffRec** **(e) Ours** **(f) GT**

Figure 4: Qualitative comparison of geometry reconstruction. Our method can recover better surface details compared to NeuS [Wang et al. 2021], PhySG [Zhang et al. 2021a], and NvDiffRec [Munkberg et al. 2022].

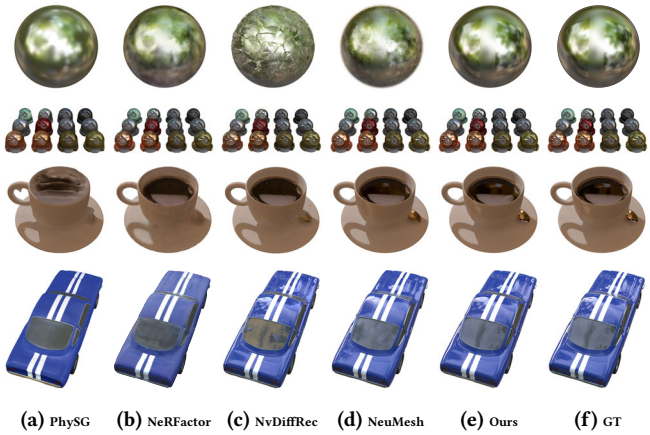
Table 1: Quantitative comparison of geometric reconstruction quality using Chamfer distance metric. All values have been mutiplied by 10 for easier reading.

Dataset	NeuS	PhySG	NvDiffRec	Ours
NeRF Synthetic	0.269	0.511	0.362	**0.266**
Shiny Blender	0.341	0.344	0.385	**0.303**

4.2 Scene Reconstruction

As shown in Fig. 4, unlike NeuS [Wang et al. 2021], PhySG [Zhang et al. 2021a] and NvDiffRec [Munkberg et al. 2022], our method avoids concave surfaces in geometry reconstruction for scenes with specular reflection by learning view-dependent and view-independent appearances separately. Quantitative results in Table 1 also show that our method outperforms these baselines.

We present novel view synthesis results in Fig. 5 and compare them with PhySG [Zhang et al. 2021a], NeRFactor [Zhang et al. 2021b], NvDiffRec [Munkberg et al. 2022], and NeuMesh [Bao et al. 2022]. PhySG fails to recover the details in the scene, due to its smooth lighting representation and its assumption that the whole scene shares the same specular BRDF material. NeRFactor uses an environment map of size 32×16 as its lighting representation, which



(a) PhySG (b) NeRFactor (c) NvDiffRec (d) NeuMesh (e) Ours (f) GT

Figure 5: Novel view synthesis comparisons with PhySG [Zhang et al. 2021a], NeRFactor [Zhang et al. 2021b], NvDiffRec [Munkberg et al. 2022], and NeuMesh [Bao et al. 2022].

Table 2: Quantitative comparison of novel view synthesis results using SSIM, PSNR, and LPIPS metrics.

Methods	NeRF Synthetic			Shiny Blender		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
PhySG	20.60	0.861	0.144	26.21	0.921	0.121
NeRFactor	27.86	0.944	0.044	27.04	0.913	0.123
NvDiffRec	29.05	0.939	0.081	28.11	0.935	0.076
NeuMesh	**30.94**	0.951	0.043	27.20	0.949	0.082
Ours	29.18	**0.959**	**0.035**	**28.79**	**0.967**	**0.072**

is unable to express sharp lighting effects. NvDiffRec may recover incorrect geometry or material with their tetrahedral representation. NeuMesh does not decompose lighting but learns the appearance with a single MLP network, which may cause wrong or blurry rendered results. Compared with these methods, our method learns decoupled appearance using two different MLP networks and uses a hybrid lighting representation so it has a better rendering quality. Quantitative comparisons are reported in Table 2.

4.3 Scene Editing

As mentioned in Sec. 3, we support editing on geometry, appearance and lighting. Our geometry editing is similar to NeuMesh [Bao et al. 2022] and we show geometry editing results in the supplementary material. In this section, we focus on the appearance and lighting editing tasks.

4.3.1 Appearance Editing. We show appearance editing comparisons with NeuMesh [Bao et al. 2022] in Fig. 8. NeuMesh renders plausible results from the editing viewpoint after optimization, but the rendered results from another viewpoint become inconsistent with the input editing. Our method optimizes the learnable features of diffuse albedo l_a to minimize the difference between rendered diffuse color and editing target using Eqn. 9 so the edited appearance matches the input editing viewed from other viewpoints and the view-dependent appearance can be preserved. We compare the image set before editing and after editing using the Frèchet Inception Distance (FID) [Heusel et al. 2017] metric to evaluate the image quality after editing in Table 3. Compared with NeuMesh, our rendered images score higher in all datasets, indicating higher image quality after editing. We show the specular and roughness editing results that NeuMesh does not support in the supplementary material.

Table 3: Quantitative comparison of appearance editing results with NeuMesh [Bao et al. 2022] using the FID metric (the lower the better).

Methods	NeRF Synthetic	Shiny Blender
NeuMesh	216.06	196.37
Ours	**194.70**	**164.73**

Table 4: Quantitative comparison of novel view synthesis results after relighting using SSIM, PSNR, and LPIPS metrics. Results are averaged over ten different viewpoints with eight different environment maps.

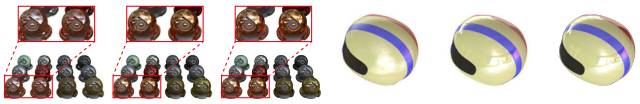
Methods	NeRF Synthetic			Shiny Blender		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
PhySG	17.56	0.722	0.0885	18.399	0.899	0.0939
InvRender	19.72	0.770	0.0780	18.690	0.908	0.0873
NeRFactor	19.35	0.815	0.0942	19.791	0.916	0.0858
NvDiffRec	19.69	**0.820**	0.0771	20.703	0.889	0.113
NvDiffRecMC	**20.09**	0.816	0.0760	21.605	0.922	0.0988
Ours	19.98	0.811	**0.0727**	**23.993**	**0.956**	**0.0409**

4.3.2 Relighting. We compare with recent PhySG [Zhang et al. 2021a], InvRender [Zhang et al. 2022b], NeRFactor [Zhang et al. 2021b], NvDiffRec [Munkberg et al. 2022], and NvDiffRecMC [Hasselgren et al. 2022] that learn to decompose geometry, material and lighting in Fig. 9. PhySG, InvRender and NeRFactor fail to express high-frequency environmental lighting due to their smooth or low-resolution lighting representations. NvDiffRec and NvDiffRecMC can handle high-frequency lighting with their high-resolution environment map but may fail to reconstruct correct geometry or material, leading to less faithful results. Our method extracts more accurate geometry and produces better relighting results with the hybrid lighting representation. We also evaluate the relighting results using PSNR, SSIM, and LPIPS metrics in Table 4 by comparing the relighting results with ground truth images generated by Blender. Overall, our relighting results have higher quality.

4.4 Ablation studies

In this subsection, we evaluate several design choices in our pipeline by conducting ablation studies on them.

4.4.1 Hybrid Lighting. We use a hybrid lighting representation of an environment map and a lighting network. To evaluate this representation, we compare it with a baseline that renders both the diffuse color and specular color using the explicit environment map with the microfacet model [Walter et al. 2007] in Fig. 6 and Table 5.



(a) Explicit (b) Hybrid (c) GT (d) Explicit (e) Hybrid (f) GT

Figure 6: Qualitative comparison of reconstruction results with the explicit lighting baseline.

Table 5: Quantitative comparison of reconstruction results with the explicit lighting baseline on the Shiny Blender dataset.

Setting	PSNR ↑	SSIM ↑	LPIPS ↓
Explicit	25.06	0.894	0.221
Hybrid	**28.79**	**0.967**	**0.072**

(a) Input (b) w/o mipmap (c) w/ mipmap (d) Input (e) w/o mipmap (f) w/ mipmap

Figure 7: Qualitative comparisons between relighting results without and with mipmap interpolation. The relit scene can better preserve the roughness of the input scene when mipmap is applied.

Table 6: Quantitative comparison of relighting results with mipmap and without mipmap on the Shiny Blender dataset.

Setting	PSNR ↑	SSIM ↑	LPIPS ↓
Without Mipmap	19.52	0.853	0.103
With Mipmap	**25.22**	**0.933**	**0.050**

The baseline struggles to reconstruct high-frequency lighting effects and our hybrid lighting representation outperforms it in terms of reconstruction quality.

4.4.2 Mipmap Relighting. We construct a mipmap of the target environment map based on the roughness values for the relighting task. We show comparisons between the relighting results with and without mipmap interpolation in Fig. 7. The relit scenes may have a mirror-like appearance if mipmap is not applied while the roughness can be well preserved when we utilize mipmap interpolation, leading to more faithful relighting results. We also evaluate it quantitatively in Table 6 and the rendered images of the relit scenes have a higher image quality when the mipmap strategy is applied.

5 DISCUSSION AND CONCLUSION

In this paper, we present a geometry, appearance and lighting editing method for neural radiance fields. The technical core is a geometry, appearance and lighting decoupling network that optimizes the learnable geometry and appearance features defined on mesh vertices, the environment map, and the specular lighting network all at once. Building upon this decoupling network, appearance editing from a given viewpoint can be seamlessly transferred to other viewpoints. In addition, our hybrid lighting representation composed of an explicit environment map and an implicit lighting network can well simulate the lighting effects in the scene and supports high-frequency environmental relighting. Nevertheless, our approach still has the following limitations: Firstly, our method does not jointly optimize the geometry in the decoupling step, which may lead to poor reconstruction of thin structures (see the first row of Fig. 8). Secondly, our method works better on relatively convex objects since it does not consider shadow or inter-reflection and produces wrong decoupled results when shadow and inter-reflection exist in the scene as shown in Fig. 10. For future exploration, we would like to learn geometry, appearance and lighting in an end-to-end manner so that all components can be optimized jointly. It is also possible to combine generative models and neural radiance field editing, for example, we can leverage newly developed diffusion models [Rombach et al. 2022] to help with image editing or apply deep geometric generative models [Gao et al. 2019] and texture generative models [Gao et al. 2021; Wang et al. 2014] to neural radiance fields.

ACKNOWLEDGMENTS

This work was supported by grants from the National Natural Science Foundation of China (No. 62061136007), the Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), and Royal Society Newton Advanced Fellowship (No. NAF\R2\192151).

REFERENCES

Chong Bao, Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. 2022. NeuMesh: Learning Disentangled Neural Mesh-based Implicit Field for Geometry and Texture Editing. In *ECCV*. 597–614.

Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David J. Kriegman, and Ravi Ramamoorthi. 2020. Deep 3D Capture: Geometry and Reflectance From Sparse Multi-View Images. In *CVPR*. 5959–5968.

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. 2021a. NeRD: Neural reflectance decomposition from image collections. In *ICCV*. 12684–12694.

Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P.A. Lensch. 2021b. Neural-PIL: Neural Pre-Integrated Lighting for Reflectance Decomposition. In *Advances in Neural Information Processing Systems*. 10691–10704.

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *CVPR*. 5939–5948.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields Without Neural Networks. In *CVPR*. 5501–5510.

Lin Gao, Tong Wu, Yu-Jie Yuan, Ming-Xian Lin, Yu-Kun Lai, and Hao Zhang. 2021. TM-NET: Deep Generative Networks for Textured Meshes. *ACM Trans. Graph.* 40, 6 (2021), 263:1–263:15.

Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. 2019. SDM-NET: deep generative network for structured deformable mesh. *ACM Trans. Graph.* 38, 6 (2019), 243:1–243:15.

Stephan J Garbin, Marek Kowalski, Virginia Estellers, Stanislaw Szymanowicz, Shideh Rezaeifar, Jingjing Shen, Matthew Johnson, and Julien Valentin. 2022. VolTeMorph: Realtime, Controllable and Generalisable Animation of Volumetric Representations. *arXiv:2208.00949* (2022).

Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. In *Advances in Neural Information Processing Systems*.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*. 6626–6637.

Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. 2022. StylizedNeRF: consistent 3D scene stylization as stylized NeRF via 2D-3D mutual learning. In *CVPR*. 18342–18352.

Rasmus Ramsbøl Jensen, Anders Lindbjerg Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. 2014. Large Scale Multi-view Stereopsis Evaluation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 406–413.

Brian Karis. 2013. Real Shading in Unreal Engine 4. (2013).

Jaroslav Krivánek and Mark Colbert. 2008. Real-time Shading with Filtered Importance Sampling. *Comput. Graph. Forum* 27, 4 (2008), 1147–1154.

Zhengfei Kuang, Kyle Olszewski, Menglei Chai, Zeng Huang, Panos Achlioptas, and Sergey Tulyakov. 2022. NeROIC: Neural Object Capture and Rendering from Online Image Collections. *Computing Research Repository (CoRR)* abs/2201.02533 (2022).

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020), 194:1–194:14.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* (2020), 15651–15663.

Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. 2021. Editing conditional radiance fields. In *ICCV*. 5773–5783.

Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. 2022. SparseNeuS: Fast Generalizable Neural Surface Reconstruction from Sparse views. *ECCV* (2022), 210–227.

William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*. 163–169.

Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. 2003. A data-driven reflectance model. *ACM Trans. Graph.* 22, 3 (2003), 759–769.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*. 4460–4470.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*. 405–421.

Jacob Munkberg, Wenzheng Chen, Jon Hasselgren, Alex Evans, Tianchang Shen, Thomas Müller, Jun Gao, and Sanja Fidler. 2022. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *CVPR*. 8270–8280.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2020. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In *CVPR*. 3501–3512.

Michael Oechsle, Lars M. Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. 2019. Texture Fields: Learning Texture Representations in Function Space. In *ICCV*. 4530–4539.

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*. 5589–5599.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*. 165–174.

Yicong Peng, Yichao Yan, Shenqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. 2022. CageNeRF: Cage-based Neural Radiance Fields for Generalized 3D Deformation and Animation. In *Advances in Neural Information Processing Systems*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *CVPR*. 10684–10695.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems*. 6087–6101.

Olga Sorkine-Hornung and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*.

Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*. 7495–7504.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: image synthesis using neural textures. *ACM Trans. Graph.* 38, 4 (2019), 66:1–66:12.

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd E. Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. In *CVPR*. 5481–5490.

Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet models for refraction through rough surfaces. In *Eurographics conference on Rendering Techniques*. 195–206.

Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. 2022. NeRF-Art: Text-Driven Neural Radiance Fields Stylization. *arXiv:2212.08070* (2022).

Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. 2009. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph.* 28, 5 (2009), 133.

Miao Wang, Yu-Kun Lai, Yuan Liang, Ralph R. Martin, and Shi-Min Hu. 2014. BigerPicture: data-driven image extrapolation using graph matching. *ACM Trans. Graph.* 33, 6 (2014), 173:1–173:13.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *Advances in Neural Information Processing Systems*. 27171–27183.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. 2022. Voxurf: Voxel-based Efficient and Accurate Neural Surface Reconstruction. *arXiv:2208.12697* (2022).

Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. 2021. Neutex: Neural texture mapping for volumetric neural rendering. In *CVPR*. 7119–7128.

Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-nerf: Point-based neural radiance fields. In *CVPR*. 5438–5448.

Tianhan Xu and Tatsuya Harada. 2022. Deforming Radiance Fields with Cages. In *ECCV*. 159–175.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume Rendering of Neural Implicit Surfaces. In *Advances in Neural Information Processing Systems*. 4805–4815.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. 2020. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *Advances in Neural Information Processing Systems*.

Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. 2022. NeRF-Editing: Geometry Editing of Neural Radiance Fields. In *CVPR*. 18332–18343.

Jingyang Zhang, Yao Yao, and Long Quan. 2021c. Learning Signed Distance Field for Multi-View Surface Reconstruction. In *ICCV*. 6525–6534.

Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. 2022a. ARF: Artistic Radiance Fields. In *ECCV*. 717–733.

Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021a. PhySG: Inverse Rendering with Spherical Gaussians for Physics-based Material Editing and Relighting. In *CVPR*. 5453–5462.

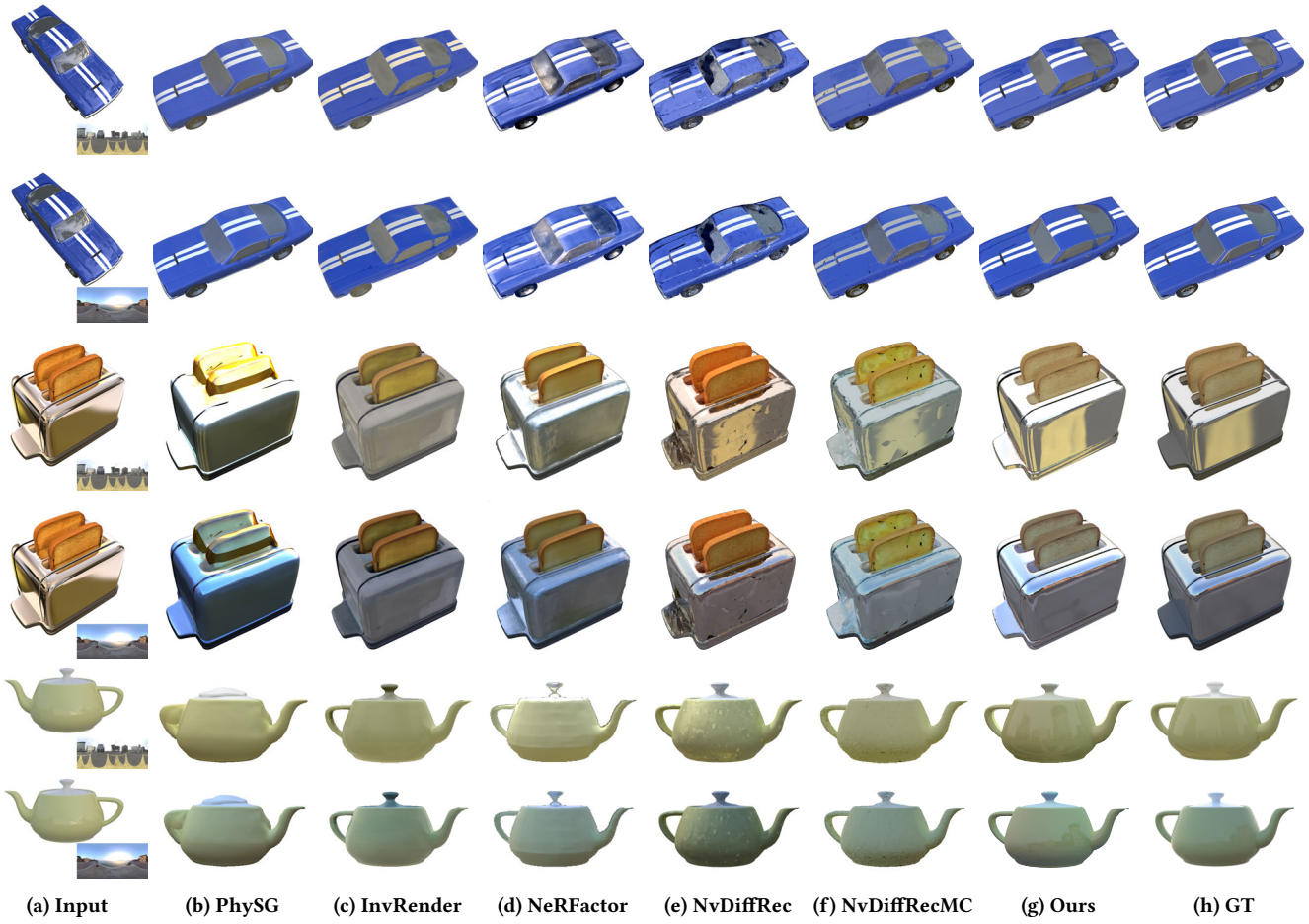
Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*. 586–595.

Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021b. NeRFactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.* 40, 6 (2021), 1–18.

Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. 2022b. Modeling Indirect Illumination for Inverse Rendering. In *CVPR*. 18622–18631.

(a) Source Image (b) Edited Image (c) NeuMesh View #1 (d) NeuMesh View #2 (e) Ours View #1 (f) Ours View #2

Figure 8: Scene appearance editing comparison with NeuMesh [Bao et al. 2022]. NeuMesh [Bao et al. 2022] can generate plausible rendering results from the editing viewpoint but rendered results from another viewpoint may be inconsistent with the input editing. Our method produces more faithful editing results from both editing viewpoint and novel viewpoints.



(a) Input (b) PhySG (c) InvRender (d) NeRFactor (e) NvDiffRec (f) NvDiffRecMC (g) Ours (h) GT

Figure 9: Scene relighting comparisons with PhySG [Zhang et al. 2021a], InvRender [Zhang et al. 2022b], NeRFactor [Zhang et al. 2021b], NvDiffRec [Munkberg et al. 2022], and NvDiffRecMC [Hasselgren et al. 2022]. In each row, the input scene and target environment map are shown in the first column. In other columns, we show relighting results by different methods and the ground truth relighting result. With the help of our reconstructed geometry and hybrid lighting representation, our method can produce more faithful relighting results with high-frequency details. We show more relighting comparisons in the supplementary material.

(a) Input (b) Diffuse Albedo (c) Diffuse Color (d) Specular Tint (e) Specular Color

Figure 10: Failure case: for an input scene with interreflections (the first row) and shadows (the second row), our decomposition network may produce wrong decomposition results and bakes them into appearance.