VD-NeRF: Visibility-aware Decoupled Neural Radiance Fields for View-Consistent Editing and High-Frequency Relighting

Tong Wu, Jia-Mu Sun, Yu-Kun Lai, and Lin Gao*.

Abstract—Neural Radiance Fields (NeRFs) 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, many approaches use Spherical Gaussian (SG) or Spherical Harmonic (SH) functions, or low-resolution environment maps to model lighting. These methods, however, struggle with high-frequency environmental relighting. While some approaches utilize high-resolution environment maps, the strategy of jointly optimizing geometry, material, and lighting introduces additional ambiguity. To solve the above problems, we propose VD-NeRF, a visibility-aware approach to decoupling 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. We further take visibility into consideration when rendering and decoupling the input 3D scene, which improves the quality of decomposition and relighting results and also enables more downstream applications such as scene composition where occlusions between scenes are common. Extensive experiments show that our method achieves better editing and relighting performance both quantitatively and qualitatively compared to previous methods.

Index Terms—Neural Radiance Fields, Inverse Rendering, Editing.

1 INTRODUCTION

Neural Radiance Fields (NeRFs) [1] 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.

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.

On the geometry editing side, a few methods propose to deform neural radiance fields by deforming sample points on a ray [2], [3], [4], [5]. 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 [6] and NeRD [7] use MLP (Multi-Layer Perceptron) networks to predict BRDF (Bidi-

* *Corresponding Author is Lin Gao (gaolin@ict.ac.cn).*
- *Tong Wu, Jia-Mu Sun, and Lin Gao are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences. Tong Wu, Jia-Mu Sun, and Lin Gao are also with University of Chinese Academy of Sciences, Beijing, China.*
  E-mail: wutong19s@ict.ac.cn, sunjiamu21@mails.ucas.ac.cn, gaolin@ict.ac.cn
- *Yu-Kun Lai is with School of Computer Science & Informatics, Cardiff University, UK.*
  E-mail: LaiY4@cardiff.ac.uk

rectional 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 [8] 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 [9] proposes not to explicitly decompose BRDF materials but instead learns view-dependent and view-independent appearance simultaneously. Although achieving high-quality reconstruction results, RefNeRF [9] 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 [10] 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 [10] is usually distorted and hard to be edited. To resolve this issue, NeuMesh [11] 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, including cases with inter-object occlusions, we propose VD-NeRF, a visibility-aware approach 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, VD-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. This paper substantially extends our previous conference paper [12] by taking visibility into account to achieve more accurate decoupling, especially when there are significant occlusions. It further enables downstream applications such as scene composition where occlusions between scenes commonly occur. We also substantially extended evaluation, including qualitative and quantitative comparisons with existing methods and further ablation study results. Our contributions can be summarized as follows:

- A neural radiance field 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.
- Our rendering considers visibility information with explicit ray-mesh intersection, which avoids shape-material ambiguity during optimization and enables more faithful decomposition.

2 Related Work

2.1 Neural Geometry Reconstruction

With the development of neural rendering [13], [14] and implicit geometry representations [15], [16], [17], *surface-based* rendering methods [18], [19], [20] 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 (NeRFs) [1], 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 [21], which derives an unbiased and occlusion-aware formulation for the neural radiance field's density function from a signed distance function (SDF). UNISURF [22] 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. [23] 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 [24] 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 [25] defines learnable features on voxel grids similar to [26], [27] to speed up training. Ref-NeuS [28] proposes to replace the radiance field with that in RefNeRF [9] to distinguish view-dependent and view-independent appearances for better geometry estimation.

2.2 NeRF Decomposition

Recently, researchers started to disentangle geometry, material and lighting from Neural Radiance Fields. NeRV [29] 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. Yuan and Fujishiro [30] approximates lighting with Spherical Gaussian (SG) functions. NeRD [7] also adopts SG as its lighting representation and reduces the learning difficulty by first extracting

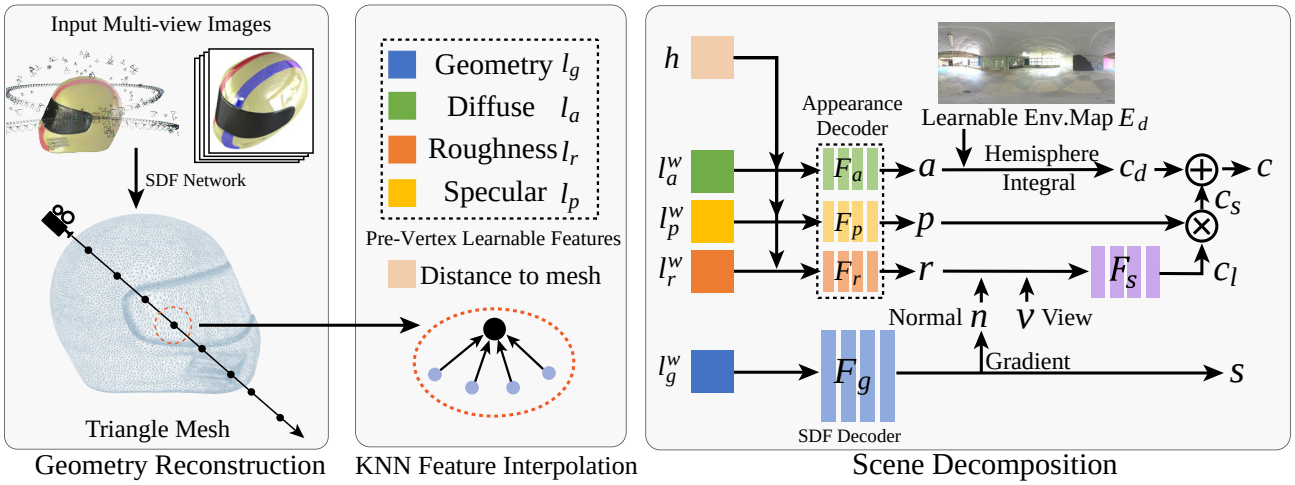


Figure 2: Given a set of images, we learn a signed distance function (SDF) to reconstruct the geometry. Then, on the vertices of the reconstructed mesh, we set up learnable geometry features l_g and appearance features l_a, l_r, l_p (corresponding to diffuse, roughness and specular components) to decompose 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, and the visibility of each sample point is determined by tracing the light right to the reconstructed mesh. 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.

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. [31] predict BRDF materials with a material autoencoder pre-trained on a BRDF material dataset [32]. NeROIC [33] approximates lighting with Spherical Harmonic (SH) coefficients and decomposes static appearance and transient appearance. NeRFactor [8] is the first work to learn shadow decomposition under unknown lighting conditions. Similar to [7], 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 [9] 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 [6] models geometry as an SDF network and its lighting is approximated by a composition of several Spherical Gaussian (SG) functions [34]. It utilizes the Disney BRDF model [35] and assumes that the scene can only have one single specular BRDF material, causing a performance drop on more complex scenes. InvRender [36] further models indirect illumination with another set of SG functions to handle more complicated appearances like inter-reflection. TensoIR [37] also models secondary bounces by querying the visibility and incident light using an implicit neural networks. NMF [38] instead models secondary effects by sampling secondary rays according to the volume rendering quadrature weight of the current primary sample point and the Trowbridge-Reitz distribution determined by the normal vector and roughness value. NvdiffRec [39] and NvdiffRecMC [40] use Deep Marching Tetrahedra [41] as its geometry representation and learn to decompose the input scene with differential rasterization rendering [42] 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 [2], [3], [5] 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 [43], sample points in the rendering process are transformed along with the mesh via barycentric coordinate interpolation. For appearance editing, several methods [44], [45], [46] propose to edit the appearance of NeRF by stylizing it with an image or text prompt. EditNeRF [47] 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 [10] 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 [11] reconstructs the geometry of the scene using NeuS [21] 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 [47]. 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 VD-NeRF, a visibility-aware decoupled geometry, appearance and lighting editing method for NeRF that allows view-consistent appearance editing and high-frequency environmental relighting. We in particular take visibility into account for more accurate decomposition. 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, as well as compositing scenes (Sec. 3.3).

3.1 Geometry Reconstruction

Recent neural implicit representations [15], [16], [17] and neural rendering techniques [1] 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 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 [21] 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 [9] 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:

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

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 [48] 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 [11] and PointNeRF [49]. $l_*^w(x)$ 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:

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

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:

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

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 source so that the diffuse color c_d for a point can be obtained by integrating all light sources in the environment map E_d at this point via:

$$c_d = \frac{a}{\pi} \int_\Omega V(\omega_i, x) L_i n \cdot \omega_i d\omega_i. \tag{4}$$

where 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. $V(\omega_i, x)$ represents the visibility of point x from the ith light direction w_i, which can be effectively calculated by ray-mesh intersection:

$$V(\omega_i, x) = \begin{cases} 1, & \|o_{\omega_i} - x\| \leq \|o_{\omega_i} - x_{\omega_i}^{int}\| \\ 0, & \|o_{\omega_i} - x\| > \|o_{\omega_i} - x_{\omega_i}^{int}\| \end{cases} \tag{5}$$

where $\|o_{\omega_i} - x\|$ is the distance from the light source o_{ω_i} to the sampled point x and $\|o_{\omega_i} - x_{\omega_i}^{int}\|$ is the distance from the light source o_{ω_i} to the intersection point $x_{\omega_i}^{int}$. To avoid infinite values, we put all light sources on a sphere with radius 1000 to stabilize the calculation.

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 [50] approximation in real-time rendering and the recent work RefNeRF [9] 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:

$$c_l = F_s(r, \cos\theta, \omega_r). \tag{6}$$

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:

$$L = L_c + L_{sdf} + \lambda_1 L_e + \lambda_2 L_{gs} + \lambda_3 L_{ec}, \tag{7}$$

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.

$$L_{sdf} = \sum_{v \in V} \sum_{i=1}^{N} \left\| s_{v,i} - s_{v,i}^t \right\|^2. \tag{8}$$

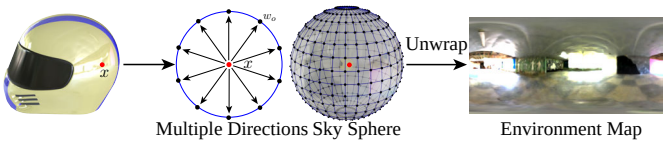


Figure 3: Given a sample point in the scene (the red point), we sample multiple directions ω_o from the sample point to points (black points on the blue frame) on the sky sphere. We treat these directions as view directions and feed them along with the roughness value of the sample point into the specular 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_{gs} is a smoothness loss that penalizes differences between adjacent vertices' geometry features and is defined as:

$$L_{gs} = \sum_i \sum_{j \in \mathcal{N}(i)} \|l_{g_i} - l_{g_j}\|_2, \tag{9}$$

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. 6 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 Split-Sum [50], 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:

$$L_{ec} = \sum_{j=1}^{P} ||F_s(0, 1, \omega_{o_j}) - E_d(\omega_{o_j})||, \tag{10}$$

where P is the number of pixels in the environment map E_d. 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 [11], we apply As-Rigid-As-Possible deformation [43] 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:

$$\arg\min_{l_*^e} \sum_{v \in V^e} ||C_\#(v) - C_e(v)||, * \in \{a, r, p\}, \quad (11)$$

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:

$$L_{relight} = \sum_{i=1}^{S} \sum_{j=1}^{P} ||F_s(0, 1, \omega_{o_j}) - E^t(\omega_{o_j})||, \quad (12)$$

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. 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. 10 that the normal direction n is the same as the view direction ω_o, so $\cos\theta = \cos 0 = 1$.

However, Eqn. 12 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. 12 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. 17). Thus, we construct a mipmap of the target environment map by computing pre-filtered environment maps at different roughness levels with pre-filter importance sampling [51]:

$$L(\omega_o) = \int_\Omega L_i(\omega_i)\,(\omega_i \cdot n)\,d\omega_i \approx \frac{\sum_{j=1}^{J} L_i(\omega_i^j)\left(\omega_i^j \cdot n\right)}{\sum_{j=1}^{J}\left(\omega_i^j \cdot n\right)} \quad (13)$$

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 [51]. 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. 12 can be further improved:

$$L_{relight} = \sum_{i=1}^{S} \sum_{j=1}^{P} ||F_s(r_i, 1, \omega_{o_j}) - M(\omega_{o_j}, r_i)|| \quad (14)$$

where M is the pre-filtered environment mipmap computed by Eqn. 13 and $M(\omega_{o_j}, r_i)$ is the light color viewed from direction ω_{o_j} and interpolated by roughness r_i.

3.3.4 Composition

Apart from editing a single 3D scene, our method also supports compositing different scenes and rendering them under a specific illumination. Taking compositing two scenes as an example, let the reconstructed meshes for these two scenes be M_a and M_b. For each sample point x on a ray, we first compare the distances from this point to the two meshes and choose the closest mesh to perform feature interpolation and decoding in Eqn. 3 to decode the signed distance value, the diffuse albedo, the specular tint, and the roughness. In terms of shading calculation, to take the occlusion between different scenes into consideration, we determine the visibility $V(\omega_i, x)$ by casting rays along the light directions onto the merged mesh of M_a and M_b. After calculating the colors of the sampled points, we also transform the signed distance values of the sampled points to opacity values and perform volume rendering to render the corresponding pixel color.

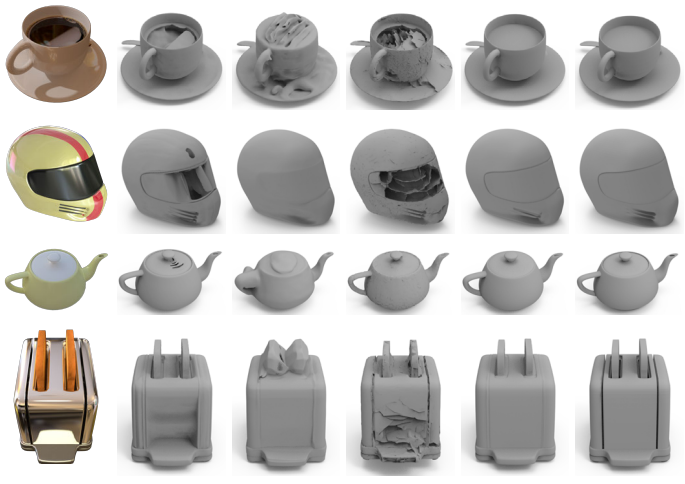
4 Results and Evaluations

4.1 Datasets and Evaluation metrics

We conduct our experiments mainly on three synthetic datasets, NeRF Synthetic [1], Shiny Blender [9], and NeILF synthetic [52] datasets. 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 [53], PSNR, and LPIPS [54] 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) [55] between the image set before editing and after editing, which has been widely used in image generation and editing tasks. In addition, we perform qualitative experiments on the real DTU [56] dataset. For training details and the network architecture, please refer to the supplementary material.

4.2 Scene Reconstruction

As shown in Fig. 4, unlike NeuS [21], PhySG [6] and NvDiffRec [39], 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.



(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 [21], PhySG [6], and NvDiffRec [39].

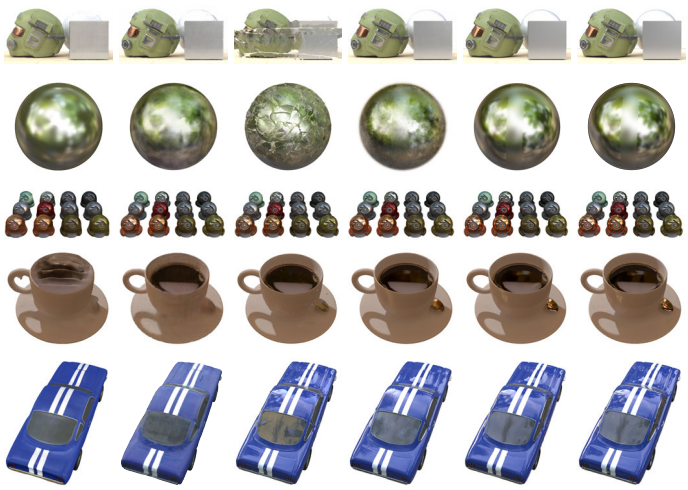
Table 1: Quantitative comparison of geometric reconstruction quality using Chamfer distance metric. All values have been mutiplied by 10 for easier reading.

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

We present novel view synthesis results in Fig. 5 and compare them with PhySG [6], NeRFactor [8], NvDiffRec [39], and NeuMesh [11]. 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 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 Decomposition

Here we show the decoupled diffuse albedo component of different scenes and compare our results with PhySG [6], NeRFactor [8], InvRender [36], NvDiffRec [39], and NvDiffRecMC [40] in Fig. 9. We also evaluate the SSIM, PSNR, and LPIPS values between the decoupled diffuse albedo and the ground truth albedo in Table 5. Both qualitative and quantitative results show that our method recovers more accurate albedo. Our full model with environment map consistency loss (L_{ec}) and visibility modeling achieves better results than those obtained by the method without either component.



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

Figure 5: Novel view synthesis comparisons with PhySG [6], NeRFactor [8], NvDiffRec [39], and NeuMesh [11].

4.4 Scene Editing

As mentioned in Sec. 3, we support editing on geometry, appearance and lighting.

4.4.1 *Geometry Editing*

Similar to NeuMesh [11], our method supports geometry editing by deforming the reconstructed mesh. We show geometry editing results in Fig. 10.

4.4.2 *Appearance Editing*

We show appearance editing comparisons with NeuMesh [11] in Fig. 6. NeuMesh renders plausible results from the editing viewpoint after optimization, but the rendered results from another viewpoint become inconsistent with the input editing. This is because NeuMesh dose not separately model light and material. Solely changing the appearance features without modifying the radiance network cannot produce faithful editing results from novel views. In addition, NeuMesh does not apply geometry smoothness loss L_{gs} on the geometry features and may produce fuzzy rendered results as shown in the 1st row of Fig. 6. The effect of this smoothness loss is visualized in Fig. 12. Our method optimizes the learnable features of diffuse albedo l_a to minimize the difference between rendered diffuse color and editing target using Eqn. 11 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) [55] 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.

4.4.3 *Relighting*

We compare with recent PhySG [6], InvRender [36], NeRFactor [8], NvDiffRec [39], and NvDiffRecMC [40] that learn to decompose geometry, material and lighting in

Table 2: Quantitative comparison of novel view synthesis results using SSIM, PSNR, and LPIPS metrics. The best and second results are highlighted with red and yellow shadings, respectively. "w/o Vis." means our method without visibility checking. "Joint" means computing the visibility by tracing the signed distance field and jointly optimizes geometry, texture, and lighting at decomposition stage.

Methods	NeRF Synthetic			Shiny Blender			NeILF Synthetic		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
PhySG	20.60	0.861	0.144	26.21	0.921	0.121	24.15	0.914	0.104
NeRFactor	27.86	0.944	0.044	27.04	0.913	0.123	27.38	0.928	0.089
NvDiffRec	29.05	0.939	0.081	28.11	0.935	0.076	19.63	0.851	0.205
NeuMesh	30.94	0.951	0.043	27.20	0.949	0.082	29.04	0.941	0.059
w/o Vis.	29.18	0.959	0.035	28.79	0.967	0.072	31.07	0.958	0.061
Joint	22.34	0.875	0.109	26.46	0.918	0.117	26.46	0.906	0.094
Ours	29.26	0.954	0.031	28.84	0.969	0.078	31.21	0.955	0.050

Table 3: Quantitative comparison of appearance editing results with NeuMesh [11] 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	18.39	0.899	0.0939
InvRender	19.72	0.770	0.0780	18.69	0.908	0.0873
NeRFactor	19.35	0.815	0.0942	19.79	0.916	0.0858
NvDiffRec	19.69	0.820	0.0771	20.70	0.889	0.113
NvDiffRecMC	20.09	0.816	0.0760	21.61	0.922	0.0988
w/o Vis.	19.98	0.811	0.0727	23.99	0.956	0.0409
Ours	20.21	0.825	0.0649	24.38	0.963	0.0537

Fig. 7. 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 by separating view-dependent and view-independent appearance learning, which provides better normal estimation that benefits the shading calculation. We further propose to use a mixture of an explicit environment map for diffuse lighting and an implicit specular lighting network, enabling high-quality relighting. 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. Note that the NeILF dataset does not provide ground truth environment maps for quantitative evaluation. We show more relighting results on the NeILF [52] Synthetic dataset and the real DTU [56] dataset in Fig. 8.

4.4.4 Composition

Our method can composite two different scenes and render them under the same lighting conditions from different viewpoints. We show rendered results of composed scenes in Fig. 11. Thanks to our visibility modeling, our method can simulate the interaction between the two composited scenes, e.g., the shadow of the chair is cast onto the hot dog in the first row of Fig. 11.

4.5 Ablation studies

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

4.5.1 Geometry Smoothness Loss

In Sec. 3, we apply a smoothness loss term to the geometry features for adjacent vertices on the reconstructed mesh. Now we evaluate its influence on the quality of the extracted meshes in Fig. 12. It shows that the geometry smoothness loss encourages a smoother surface reconstruction. We also evaluate this loss with quantitative Chamfer Distance results in Table 6, which shows the reconstructed mesh is closer to the ground truth when this loss is applied.

4.5.2 Environment Consistency Loss

In Sec. 3, we separate the learning of the diffuse lighting and the specular lighting and constrain them to be consistent with an environment consistency loss L_{ec}. Here we ablate how this loss influences the decoupled results in Fig. 13. When L_{ec} is not applied, the learned albedo component may present unevenly distributed colors to make up for the learned environment map with inaccurate intensity or color. Quantitative results in Table 5 also show that the environment consistency loss can improve the quality of the decoupled albedo component.

4.5.3 Visibility Modeling

In Eqn. 4, we consider the visibility of each sample point by explicitly calculating the intersection of a light source and the reconstructed mesh. We ablate the necessity of the visibility modeling in Fig. 14, which shows if we do not consider visibility in the decoupling and rendering process, the decoupled albedo may bake more shadow in the occluded region. Quantitative evaluation in Table 5 and Table 4 also indicates that visibility modeling can benefit the decoupling and relighting performances.

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

Figure 6: Scene appearance editing comparison with NeuMesh [11]. NeuMesh [11] 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.

4.5.4 Ray-Mesh Intersection Visibility

As mentioned in Sec. 3, we first reconstruct the geometry of the input scene and compute the visibility term in the rendering equation with the ray-mesh intersection for the texture and lighting optimization. Another possible solution for visibility modeling is to approximate it with transmittance as done in NeRV [29] and NeRFactor [8]. To evaluate the rationality of our ray-mesh intersection visibility, we introduce another baseline (Joint Optimization) that computes visibility by tracing the signed distance fields in the scene decomposition stage and allows joint optimization of the geometry, material, and lighting at the decomposition stage. We show reconstruction results and decomposed normals by this baseline in Fig. 15. The joint optimization at the decomposition stage in the baseline makes the inverse rendering task more ambiguous and leads to less faithful geometry and fuzzy appearance reconstruction. Our ray-mesh intersection visibility is not perfectly accurate (e.g., it might miss some thin structure) but still alleviates the shape-appearance ambiguity and allows more accurate reconstruction and geometry decomposition. Quantitative results of reconstruction by this baseline can be found in the "Joint" row of Table 2 and Table 5.

4.5.5 Hybrid Lighting

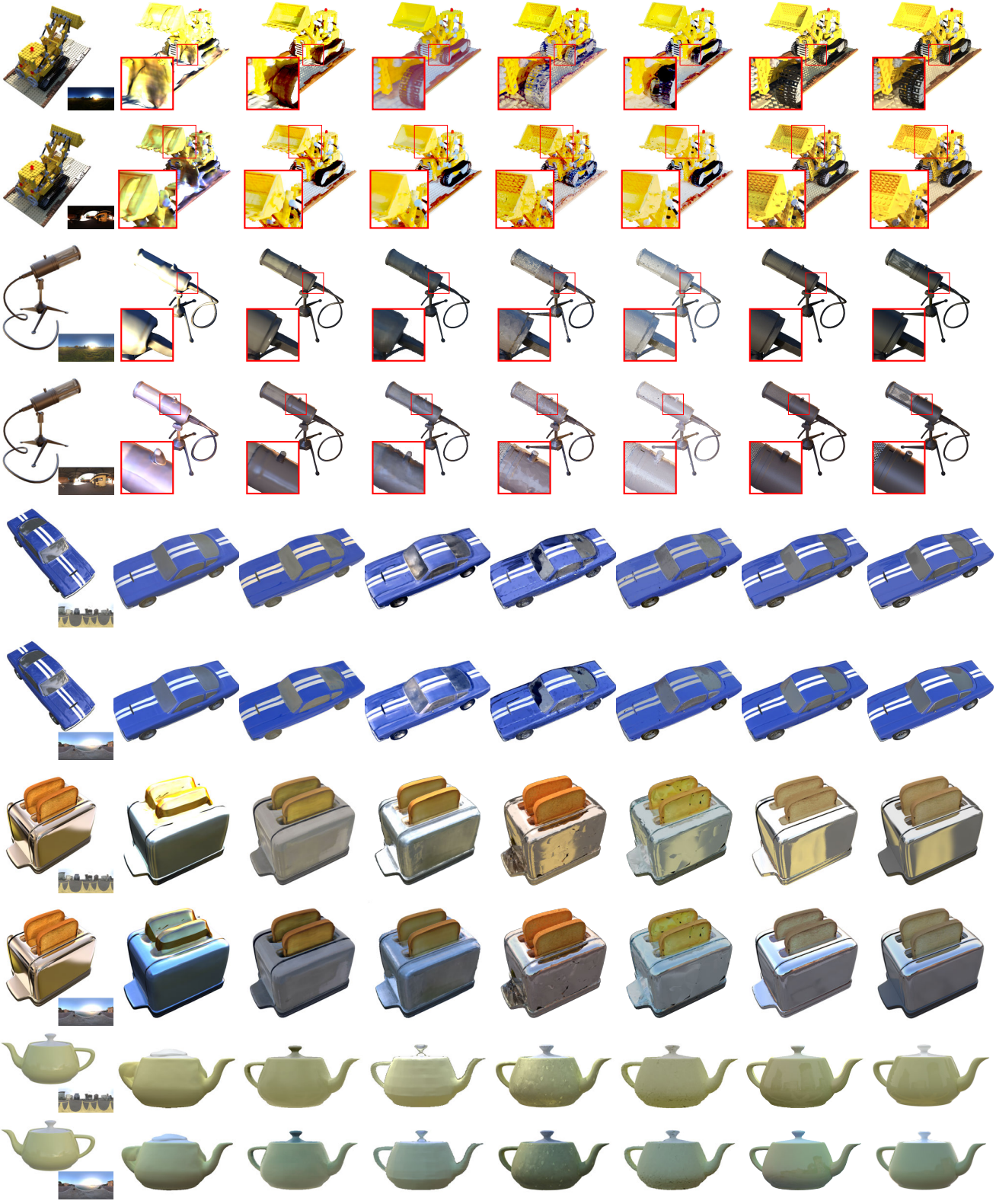
We use a hybrid lighting representation of an environment map and a lighting network. To evaluate this representation, we compare it with a baseline that renders both the diffuse color and specular color using the explicit environment map with the microfacet model [57] in Fig. 16 and Table 7. The baseline struggles to reconstruct high-frequency lighting effects and our hybrid lighting representation outperforms it in terms of reconstruction quality.

4.5.6 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. 17. 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 8 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

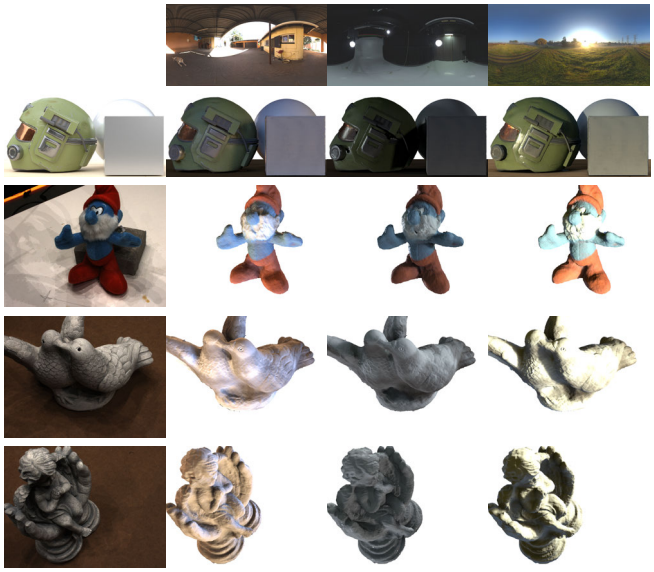


(a) Input (b) PhySG (c) InvRender (d) NeRFactor (e) NvDiffRec (f) NvDiffRecMC (g) Ours (h) GT

Figure 7: Scene relighting comparisons with PhySG [6], InvRender [36], NeRFactor [8], NvDiffRec [39], and NvDiffRecMC [40]. 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.

Table 5: Quantitative comparison of the decomposed albedo component using SSIM, PSNR, and LPIPS metrics. Results are averaged over ten different viewpoints. The best and second results are highlighted with red and yellow shadings, respectively.

Methods	NeRF Synthetic			Shiny Blender			NeILF Synthetic		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
PhySG	16.92	0.869	0.0945	21.55	0.849	0.132	16.70	0.860	0.161
InvRender	18.20	0.855	0.0914	19.96	0.850	0.129	19.12	0.893	0.113
NeRFactor	19.04	0.871	0.0867	20.28	0.884	0.106	17.34	0.858	0.132
NvDiffRec	19.06	0.848	0.209	19.66	0.880	0.209	14.14	0.778	0.219
NvDiffRecMC	18.53	0.875	0.176	19.80	0.855	0.125	15.70	0.860	0.162
w/o L_{ec}	18.31	0.845	0.125	22.69	0.903	0.0976	19.04	0.880	0.124
w/o Vis.	19.58	0.863	0.107	24.74	0.926	0.0767	17.83	0.871	0.152
Joint	17.30	0.868	0.0923	20.14	0.835	0.137	18.97	0.878	0.142
Ours	19.87	0.881	0.0892	24.81	0.928	0.0694	21.56	0.907	0.076



(a) Input (b) Relit #1 (c) Relit #2 (d) Relit #3

Figure 8: Qualitative relighting results on the NeILF [52] synthetic dataset and the real DTU [56] dataset.

Table 6: Quantitative comparison of geometric reconstruction quality using Chamfer distance metric with the baseline that does not use the geometry smoothness loss (w/o L_{gs}). All values have been mutiplied by 10 for easier reading.

Dataset	NeRF Synthetic	Shiny Blender
w/o L_{gs}	0.312	0.367
w/ L_{gs}	**0.266**	**0.303**

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

Table 7: 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	**28.84**	**0.969**	**0.078**

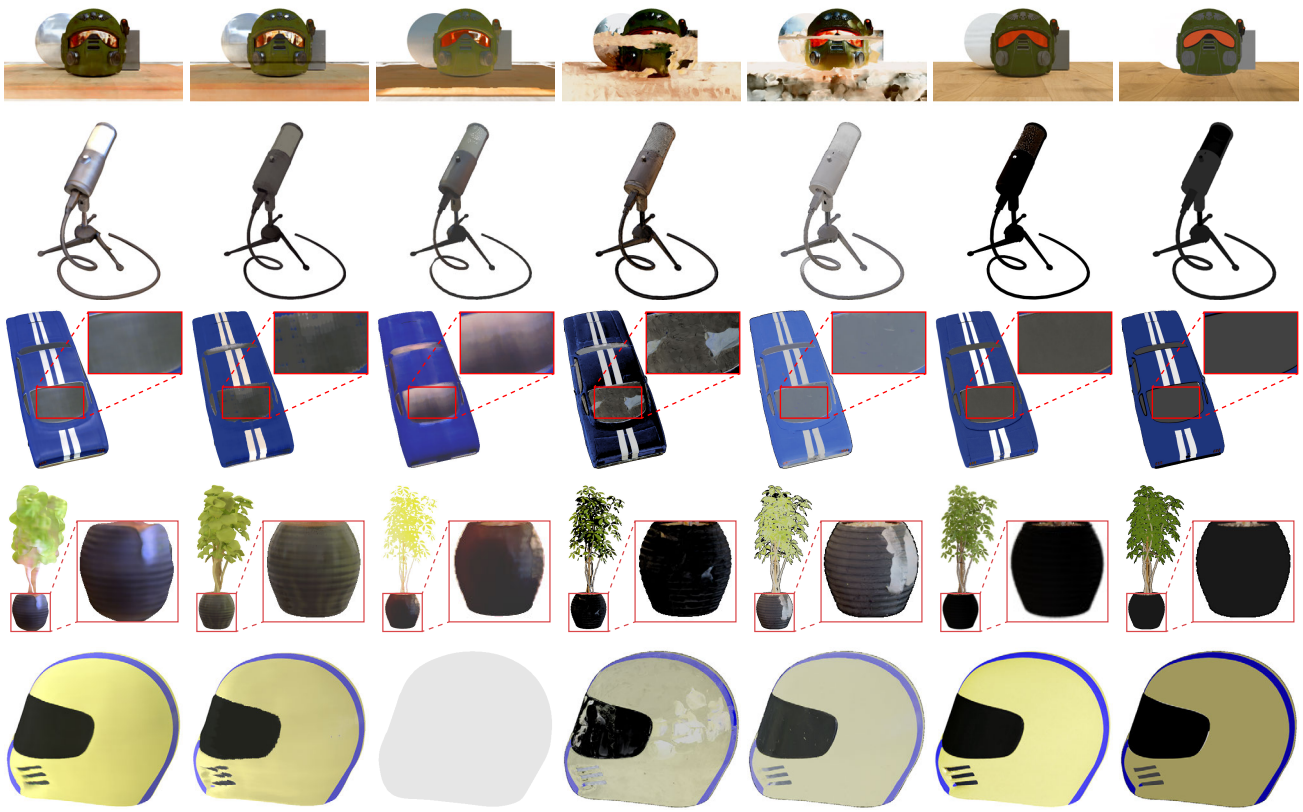
Table 8: Quantitative comparison of relighting results with mipmap and without mipmap on the Shiny Blender dataset.

Setting	PSNR ↑	SSIM ↑	LPIPS ↓
w/o Mipmap	19.52	0.853	0.103
w/ Mipmap	**24.38**	**0.963**	**0.0537**

jointly optimize the geometry in the decoupling step, which may fail to reconstruct thin structures (see the first row of Fig. 6). Secondly, our method does not consider inter-reflection in rendering and decomposition and may produce wrong decoupled results (see the first row of Fig. 18). Finally, even though we model visibility in the decoupling process, it is still a challenging task to decompose shadows and infer material parameters in the occluded regions given captured images of the 3D scene under a single fixed illumination. The introduced visibility can only alleviate shadow baking effects but is unable to fully decompose shadow and may still bake shadow into appearance as shown in the second row of Fig. 18. For future exploration, we would like to consider more complicated lighting effects like shadows and inter-reflections to get more accurate decomposition results. In addition, it is also possible to combine generative models and neural radiance field editing. For example, we can utilize recent diffusion models [58] to help with image editing or generate textures on neural radiance fields similar to BiggerPicture [59], TM-Net [60], and DiffMat [61].

ACKNOWLEDGMENTS

This work was supported by grants from the National Natural Science Foundation of China (No. 62061136007 and No. 62322210), the Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), Royal Society Newton Advanced Fellowship (No. NAF\R2\192151), and Beijing Municipal Science and Technology Commission (No. Z231100005923031).



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

Figure 9: Albedo decomposition comparisons with PhySG [6], InvRender [36], NeRFactor [8], NvDiffRec [39], and NvDiffRecMC [40]. In each row, we show the decoupled albedo components by different methods and the ground truth albedo. With the help of our reconstructed geometry and hybrid lighting representation, our method can produce decomposition results closer to the ground truth.

(a) Input (b) Edited (c) View #1 (d) View #2 (e) View #3 (f) View #4

Figure 10: We perform editing on the geometry of the input scene by manipulating the reconstructed mesh. In each row, the input geometry and the geometry after editing are shown in the first two columns and we show novel view synthesis results from four different viewpoints.

(a) Scene #1 (b) Scene #2 (c) Comp. #1 (d) Comp. #2 (e) Comp. #3 (f) Comp. #4

Figure 11: Scene composition results by our method. In each row, we show two input scenes where each composite scene is composed from and rendered results of the composed scenes from four different viewpoints. Note that the composed scenes are rendered under a novel illumination.

REFERENCES

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020, pp. 405–421.

[2] Y.-J. Yuan, Y.-T. Sun, Y.-K. Lai, Y. Ma, R. Jia, and L. Gao, "Nerf-editing: Geometry editing of neural radiance fields," in *CVPR*, 2022, pp. 18 332–18 343.

[3] T. Xu and T. Harada, "Deforming radiance fields with cages," in *ECCV*, 2022, pp. 159–175.

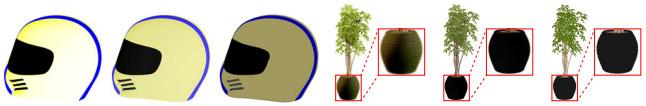
[4] Y. Peng, Y. Yan, S. Liu, Y. Cheng, S. Guan, B. Pan, G. Zhai, and X. Yang, "Cagenerf: Cage-based neural radiance fields for generalized 3d deformation and animation," in *Advances in Neural Information Processing Systems*, 2022.

[5] S. J. Garbin, M. Kowalski, V. Estellers, S. Szymanowicz, S. Rezaeifar, J. Shen, M. Johnson, and J. Valentin, "Voltemorph: Realtime, controllable and generalisable animation of volumetric representations," *arXiv:2208.00949*, 2022.

[6] K. Zhang, F. Luan, Q. Wang, K. Bala, and N. Snavely, "PhySG:

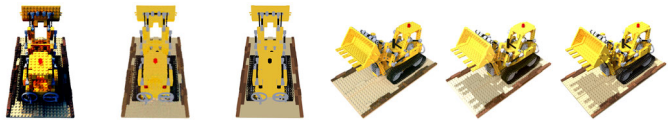
(a) Input (b) w/o L_{gs} (c) w/ L_{gs} (d) Input (e) w/o L_{gs} (f) w/ L_{gs}

Figure 12: Qualitative comparison without and with the geometry smoothness loss L_{gs}. Extracted meshes with L_{gs} applied have smoother geometry.



(a) w/o L_{ec} (b) w/ L_{ec} (c) GT (d) w/o L_{ec} (e) w/ L_{ec} (f) GT

Figure 13: Qualitative comparison of the decoupled albedo component without and with the environment consistency loss L_{ec}.



(a) w/o Vis. (b) w/ Vis. (c) GT (d) w/o Vis. (e) w/ Vis. (f) GT

Figure 14: Qualitative comparison of decoupled diffuse albedo (a)-(c) and relighting results (d)-(f) with the baseline (w/o Vis.) that does not consider visibility.

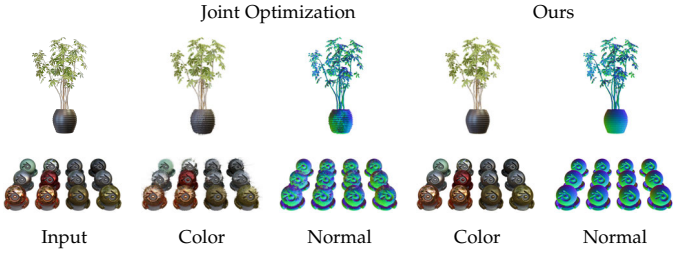


Figure 15: Qualitative comparison of our method and the baseline jointly optimizing geometry, texture, and lighting by tracing the implicit signed distance fields to get the visibility.

Inverse rendering with spherical gaussians for physics-based material editing and relighting," in *CVPR*, 2021, pp. 5453–5462.

[7] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "NeRD: Neural reflectance decomposition from image collections," in *ICCV*, 2021, pp. 12 684–12 694.

[8] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "NeRFactor: Neural factorization of shape and reflectance under an unknown illumination," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–18, 2021.

[9] D. Verbin, P. Hedman, B. Mildenhall, T. E. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-nerf: Structured view-dependent appearance for neural radiance fields," in *CVPR*, 2022, pp. 5481–5490.

[10] F. Xiang, Z. Xu, M. Hasan, Y. Hold-Geoffroy, K. Sunkavalli, and H. Su, "Neutex: Neural texture mapping for volumetric neural rendering," in *CVPR*, 2021, pp. 7119–7128.

[11] C. Bao, B. Yang, Z. Junyi, B. Hujun, Z. Yinda, C. Zhaopeng, and Z. Guofeng, "Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing," in *ECCV*, 2022, pp. 597–614.

[12] T. Wu, J.-M. Sun, Y.-K. Lai, and L. Gao, "DE-NeRF: Decoupled neural radiance fields for view-consistent appearance editing and high-frequency environmental relighting," in *ACM SIGGRAPH Conference Proceedings*, 2023.

[13] M. Oechsle, L. M. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger, "Texture fields: Learning texture representations in function space," in *ICCV*, 2019, pp. 4530–4539.

[14] J. Thies, M. Zollhöfer, and M. Nießner, "Deferred neural rendering: image synthesis using neural textures," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 66:1–66:12, 2019.

[15] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *CVPR*, 2019, pp. 5939–5948.

[16] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019, pp. 165–174.

[17] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019, pp. 4460–4470.

[18] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *CVPR*, 2020, pp. 3501–3512.

[19] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, R. Basri, and Y. Lipman, "Multiview neural surface reconstruction by disentangling geometry and appearance," in *Advances in Neural Information Processing Systems*, 2020.

[20] J. Zhang, Y. Yao, and L. Quan, "Learning signed distance field for multi-view surface reconstruction," in *ICCV*, 2021, pp. 6525–6534.

[21] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," in *Advances in Neural Information Processing Systems*, 2021, pp. 27 171–27 183.

[22] M. Oechsle, S. Peng, and A. Geiger, "Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *ICCV*, 2021, pp. 5589–5599.

[23] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," in *Advances in Neural Information Processing Systems*, 2021, pp. 4805–4815.

[24] X. Long, C. Lin, P. Wang, T. Komura, and W. Wang, "Sparseneus: Fast generalizable neural surface reconstruction from sparse views," *ECCV*, pp. 210–227, 2022.

[25] T. Wu, J. Wang, X. Pan, X. Xu, C. Theobalt, Z. Liu, and D. Lin, "Voxurf: Voxel-based efficient and accurate neural surface reconstruction," *arXiv:2208.12697*, 2022.

[26] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, pp. 15 651–15 663, 2020.

[27] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *CVPR*, 2022, pp. 5501–5510.

[28] W. Ge, T. Hu, H. Zhao, S. Liu, and Y. Chen, "Ref-neus: Ambiguity-reduced neural implicit surface learning for multi-view reconstruction with reflection," in *IEEE/CVF International Conference on Computer Vision, ICCV 2023*, 2023, pp. 4228–4237.

[29] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "NeRV: Neural reflectance and visibility fields for relighting and view synthesis," in *CVPR*, 2021, pp. 7495–7504.

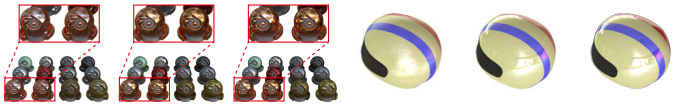
[30] L. Yuan and I. Fujishiro, "Multiview svbrdf capture from unified shape and illumination," *Visual Informatics*, vol. 7, no. 3, pp. 11–21, 2023.

[31] M. Boss, V. Jampani, R. Braun, C. Liu, J. T. Barron, and H. P. Lensch, "Neural-pil: Neural pre-integrated lighting for reflectance decomposition," in *Advances in Neural Information Processing Systems*, 2021, pp. 10 691–10 704.

[32] W. Matusik, H. Pfister, M. Brand, and L. McMillan, "A data-driven reflectance model," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 759–769, 2003.

[33] Z. Kuang, K. Olszewski, M. Chai, Z. Huang, P. Achlioptas, and S. Tulyakov, "NeROIC: Neural object capture and rendering from online image collections," *Computing Research Repository (CoRR)*, vol. abs/2201.02533, 2022.

[34] J. Wang, P. Ren, M. Gong, J. Snyder, and B. Guo, "All-frequency



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

Figure 16: Qualitative comparison of reconstruction results with the explicit lighting baseline.

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

Figure 17: 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.

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

Figure 18: Failure cases: for an input scene with interreflections (the first row) and shadows (the second row), our decomposition network may produce wrong decomposition results and bake them into appearance.

rendering of dynamic, spatially-varying reflectance," *ACM Trans. Graph.*, vol. 28, no. 5, p. 133, 2009.

[35] S. Bi, Z. Xu, K. Sunkavalli, D. J. Kriegman, and R. Ramamoorthi, "Deep 3d capture: Geometry and reflectance from sparse multi-view images," in *CVPR*, 2020, pp. 5959–5968.

[36] Y. Zhang, J. Sun, X. He, H. Fu, R. Jia, and X. Zhou, "Modeling indirect illumination for inverse rendering," in *CVPR*, 2022, pp. 18 622–18 631.

[37] H. Jin, I. Liu, P. Xu, X. Zhang, S. Han, S. Bi, X. Zhou, Z. Xu, and H. Su, "Tensoir: Tensorial inverse rendering," in *CVPR*, 2023.

[38] A. Mai, D. Verbin, F. Kuester, and S. Fridovich-Keil, "Neural microfacet fields for inverse rendering," 2023.

[39] J. Munkberg, W. Chen, J. Hasselgren, A. Evans, T. Shen, T. Müller, J. Gao, and S. Fidler, "Extracting triangular 3d models, materials, and lighting from images," in *CVPR*, 2022, pp. 8270–8280.

[40] J. Hasselgren, N. Hofmann, and J. Munkberg, "Shape, light, and material decomposition from images using monte carlo rendering and denoising," in *Advances in Neural Information Processing Systems*, 2022.

[41] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler, "Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis," in *Advances in Neural Information Processing Systems*, 2021, pp. 6087–6101.

[42] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila, "Modular primitives for high-performance differentiable rendering," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 194:1–194:14, 2020.

[43] O. Sorkine-Hornung and M. Alexa, "As-rigid-as-possible surface modeling," in *Symposium on Geometry Processing*, 2007.

[44] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao, "Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning," in *CVPR*, 2022, pp. 18 342–18 352.

[45] K. Zhang, N. Kolkin, S. Bi, F. Luan, Z. Xu, E. Shechtman, and N. Snavely, "Arf: Artistic radiance fields," in *ECCV*, 2022, pp. 717–733.

[46] C. Wang, R. Jiang, M. Chai, M. He, D. Chen, and J. Liao, "Nerf-art: Text-driven neural radiance fields stylization," *arXiv:2212.08070*, 2022.

[47] S. Liu, X. Zhang, Z. Zhang, R. Zhang, J.-Y. Zhu, and B. Russell, "Editing conditional radiance fields," in *ICCV*, 2021, pp. 5773–5783.

[48] W. E. Lorensen and H. 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*, 1987, pp. 163–169.

[49] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-nerf: Point-based neural radiance fields," in *CVPR*, 2022, pp. 5438–5448.

[50] B. Karis, "Real shading in unreal engine 4," 2013.

[51] J. Krivánek and M. Colbert, "Real-time shading with filtered importance sampling," *Comput. Graph. Forum*, vol. 27, no. 4, pp. 1147–1154, 2008.

[52] Y. Yao, J. Zhang, J. Liu, Y. Qu, T. Fang, D. McKinnon, Y. Tsin, and L. Quan, "Neilf: Neural incident light field for physically-based material estimation," in *European Conference on Computer Vision (ECCV)*, 2022.

[53] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[54] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018, pp. 586–595.

[55] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[56] R. R. Jensen, A. L. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large scale multi-view stereopsis evaluation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2014, pp. 406–413.

[57] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance, "Microfacet models for refraction through rough surfaces," in *Eurographics conference on Rendering Techniques*, 2007, pp. 195–206.

[58] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022, pp. 10 684–10 695.

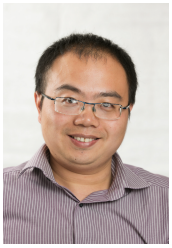
[59] M. Wang, Y. Lai, Y. Liang, R. R. Martin, and S. Hu, "Biggerpicture: data-driven image extrapolation using graph matching," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 173:1–173:13, 2014.

[60] L. Gao, T. Wu, Y.-J. Yuan, M.-X. Lin, Y.-K. Lai, and H. Zhang, "Tm-net: Deep generative networks for textured meshes," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 263:1–263:15, 2021.

[61] L. Yuan, D. Yan, S. Saito, and I. Fujishiro, "Diffmat: Latent diffusion models for image-guided material generation," *Visual Informatics*, vol. 8, no. 1, pp. 6–14, 2024.

Tong Wu received his bachelor's degree in computer science from Huazhong University of Science and Technology in 2019. He is currently a PhD candidate at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics and computer vision.

Jia-Mu Sun is a Master student of Computer Science at Institute of Computing Technology, Chinese Academy of Sciences. He received his bachelor's degree from Huazhong University of Science and Technology. His research interests include geometry learning and rendering.



Yu-Kun Lai received his bachelor's degree and PhD degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a Professor in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial boards of *IEEE Transactions on Visualization and Computer Graphics* and *The Visual Computer*.

Lin Gao received the bachelor's degree in mathematics from Sichuan University and the PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded the Newton Advanced Fellowship from the Royal Society and the Asia Graphics Association young researcher award. His research interests include computer graphics and geometric processing.