REFRAME: Reflective Surface Real-Time Rendering for Mobile Devices

Chaojie Ji[1], Yufeng Li[2], and Yiyi Liao[1]✉

[1] Zhejiang University
[2] University of Electronic Science and Technology of China

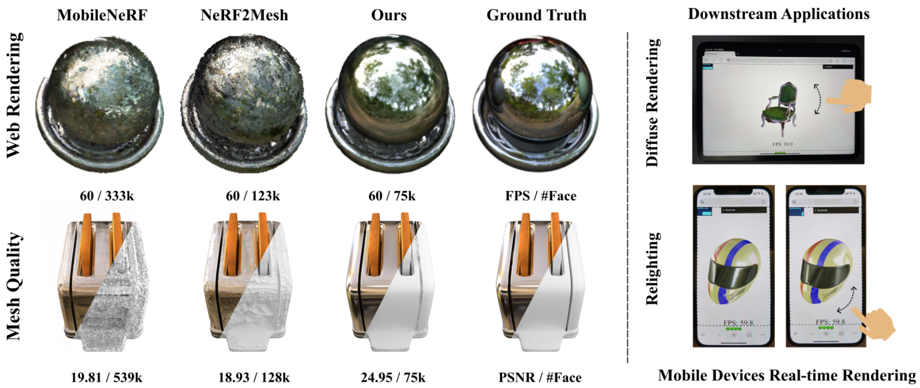


Fig. 1: REFRAME enables real-time rendering on consumer GPUs and mobile devices, delivering superior subjective quality with a low face number compared to the baselines [9, 45]. Additionally, it effectively decouples the appearance properties and environmental information of the scene, which helps its capabilities for downstream scene editing tasks.

Abstract. This work tackles the challenging task of achieving real-time novel view synthesis for reflective surfaces across various scenes. Existing real-time rendering methods, especially those based on meshes, often have subpar performance in modeling surfaces with rich view-dependent appearances. Our key idea lies in leveraging meshes for rendering acceleration while incorporating a novel approach to parameterize view-dependent information. We decompose the color into diffuse and specular, and model the specular color in the reflected direction based on a neural environment map. Our experiments demonstrate that our method achieves comparable reconstruction quality for highly reflective surfaces compared to state-of-the-art offline methods, while also efficiently enabling real-time rendering on edge devices such as smartphones. Our project page is at https://xdimlab.github.io/REFRAME/.

Keywords: Reflective surface · Real-time rendering · Mobile device

✉ Corresponding author.

1 Introduction

Novel view synthesis (NVS) generates realistic images from novel viewpoints using multiple input views. While Neural Radiance Fields (NeRF) [33] excel at high-quality NVS through volume rendering, they struggle with modeling reflective appearance and lack real-time rendering capabilities.

Several methods [19, 52, 59, 63] extend NeRF to decouple the intrinsic scene properties, *e.g.*, into materials and lighting, and obtain color by the rendering equation [20]. Decoupling the environmental lighting and the physical parameters of the object often helps in modeling reflective objects. In contrast to recovering the precise physical meaning which may harm the visual quality due to the approximated rendering equation, another line of works [14, 46, 56] avoids full decomposition but enables representing reflective objects by modeling the reflective radiance. However, none of the aforementioned methods is capable of real-time rendering, especially on mobile devices, due to the expensive query of the rendering equation, or the underlying volume rendering formulation.

There are many existing methods focusing on accelerating the rendering of NeRF. One line of works sticks with the volume rendering pipeline where the color of a pixel is accumulated along the ray. Acceleration is achieved by improving the sampling strategy [25, 27, 31, 48], tabulating the intermediate output [12, 16, 18, 34, 40, 49, 55, 58], or leveraging super-resolution neural rendering [28]. However, deploying these methods on edge devices such as smartphones is often greatly hindered due to the demanding computational power they typically require. Besides, tabulation-based methods require large memory consumption and greatly increase communication costs when transmitting data between the cloud server and the client. Another line of works [9, 39, 45] distills radiance fields into a mesh for real-time rendering, combined with a small MLP to model view-dependent effects. Mesh-based methods can leverage traditional graphics pipelines for acceleration, enabling them to achieve real-time rendering even on edge devices. Nonetheless, real-time rendering methods [9, 15, 16, 45, 58] often struggle to model objects with highly reflective surfaces. Besides, these mesh-based real-time rendering methods [9, 45, 57] typically require a large number of vertices and faces to achieve high fidelity.

In this paper, we propose **REFRAME**, a mesh-based **REF**lective surface **R**e**A**l-time rendering method for **M**obil**E** devices (*e.g.*, smartphones), see Fig. 1. We find the fact that existing NeRF distilled mesh rendering pipelines [9, 45] do not perform well in rendering objects with highly reflective appearances mainly for two reasons. Firstly, these methods utilize the viewing direction to model the view-dependent appearance, approved to be less effective than using the reflection direction [46]. As such, our method adopts the reflection direction-based parameterization. Nevertheless, this parameterization requires surface normal, yet estimating precise vertex normal based on a mesh representation can be challenging. Therefore, we propose a geometry learner that learns vertex and normal offsets through two networks. This leads to good normal estimations to calculate the reflection direction while maintaining relatively low vertex and face numbers. Secondly and more importantly, to alleviate inference burden, real-time rendering methods often have limited capacity to model complex view-dependent information. In order to enhance the expressive power of the model without increasing the inference computational cost, we employ a four-layer MLP during training to learn the environmental features. This information is then baked into a two-dimensional environment feature map during inference. Remarkably, the distilled environment feature map incurs a memory overhead of less than 1MB and can be edited for relighting purposes. Finally, despite our method achieving real-time rendering with an advantage in mesh faces and vertices number compared to existing works, the reconstruction quality of our method is comparable to the current state-of-the-art (including non-real-time methods) work and even outperforms them in rendering highly reflective objects. We summarize our contributions as follows:

- We propose a novel mesh geometry learner, allowing for robustly optimizing mesh vertex positions and normals. This leads to high rendering quality with relatively low mesh vertex and face numbers.
- We propose to use an environment feature map to model view-dependent appearances of highly reflective objects, which enhances the capacity to reconstruct complex reflective appearances without increasing the inference burden. This further enables relighting effects.
- Our rendering quality is on par with the current state-of-the-art methods while being able to achieve real-time rendering across various platform devices. Moreover, our method even surpasses the current non-real-time state-of-the-art approaches in rendering objects with highly reflective appearances.

2 Related Work

NeRF-based Scene Representation: NeRF [33] and its derivative works [1–3,41,44,46,53] have employed ray-marching based volume rendering methods to achieve high-quality and realistic rendering of different types of objects in various environments. Many follow-up works have extended NeRF in different aspects, including dynamic scene modeling [36–38], 3D-aware generation [7,8,42], and semantic scene understanding [11,24,65]. In this work, we focus on addressing two limitations of NeRF, *i.e.*, modeling of reflective objects and real-time rendering.
Reflectance Decomposition: A number of works [4, 5, 19, 43, 52, 59–64] investigate the task of inferring geometry and material properties based on neural fields representation, typically by formulating the image generation using the physically-based rendering. Despite achieving lighting and material control, these methods are typically inferior to state-of-the-art NeRF-based methods that directly model the radiance in terms of the rendering quality. This is due to the fact that these methods rely on simplified rendering equations of the real world. Another line of works achieves better rendering quality by avoiding full decomposition, yet allowing for modeling glossy objects by modeling the reflected radiance [14, 46, 56] or replacing the explicit rendering equation with a learned neural renderer [6, 30]. However, none of these methods are capable of real-time rendering due to the underlying volume rendering formulation. While

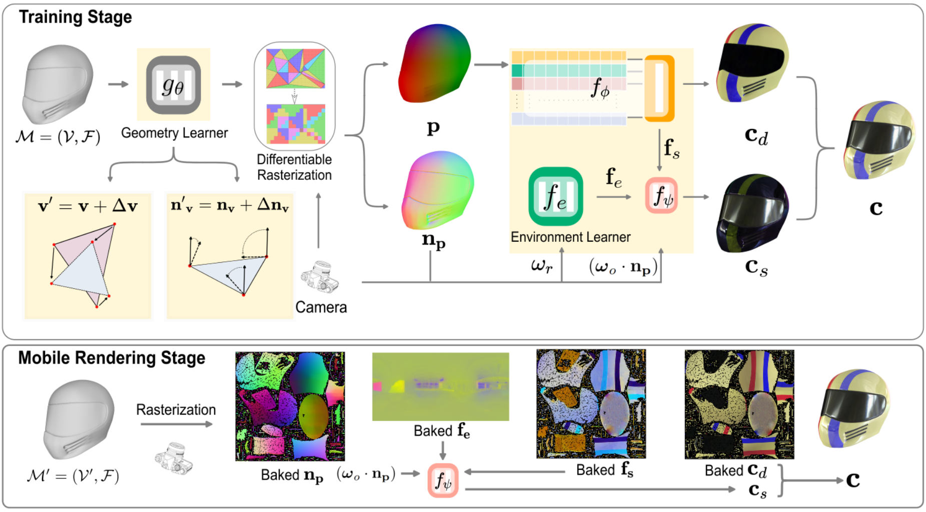


Fig. 2: Pipeline for REFRAME. Components with a yellow background are either baked or omitted during the mobile rendering stage. **Training Stage**: The initial mesh is updated first before performing differentiable rasterization [26]. Next, we obtain the diffuse color $\mathbf{c}_d$ and specular feature $\mathbf{f}_s$ based on the position $\mathbf{p}$, and the environment feature $\mathbf{f}_e$ based on reflective direction $\boldsymbol{\omega}_r$. Then we obtain the specular color $\mathbf{c}_s$ and combine it with $\mathbf{c}_d$ to create the final full color $\mathbf{c}$. **Mobile Rendering Stage**: We bake the intermediate output for real-time rendering. This mesh-based rendering can be implemented using WebGL and easily deployed on various platforms (*e.g.*, desktop and mobile devices). Here, we retrieve the $\mathbf{c}_d$ and $\mathbf{f}_s$ from baked texture images, and the $\mathbf{f}_e$ from the environment feature map. $\mathbf{c}_d$ and $\mathbf{c}_s$ are processed the same as the training stage to obtain $\mathbf{c}$.

NvDiffRec [35] enables real-time rendering based on the mesh representation, its quality is also limited by the full decomposition. In this work, we follow the partial decomposition pipeline [46] and propose to learn a neural environment map based on a NeRF-distilled mesh representation, with a focus on achieving real-time rendering.

NeRF Acceleration: Accelerating NeRF rendering is another significant research area, with approaches reducing sampling points along the ray [25, 31, 48], tabulating the intermediate output [12, 16, 18, 49, 55, 58], using thousands of small MLPs to represent the scene [10, 40], or utilizing super-resolution techniques [28, 51]. However, these methods typically require consumer GPUs for real-time rendering, making it challenging to deploy them on edge devices with limited computational resources.

More recently, a few approaches [9, 45, 57] propose to distill a NeRF representation into a mesh for real-time rendering, where MobileNeRF [9] and NeRF2Mesh [45] enable real-time rendering on mobile devices. However, MobileNeRF [9] does not model the accurate geometry of the scene, leading to expensive memory costs with a large number of vertices and faces. NeRF2Mesh [45] is the most relevant work to ours, which distills grid-based representation into a mesh. The good geometry prior for NeRF2Mesh [45] leads to a relatively low number of vertices and faces. However, none of the aforementioned methods are capable of modeling highly reflective objects faithfully due to their way of view-dependent color modeling. We propose to use a neural environment map to tackle this challenge by modeling the reflected radiance. Furthermore, we propose to estimate a high-quality vertex normal, further reducing the number of faces and vertices while maintaining the quality.

3 Method

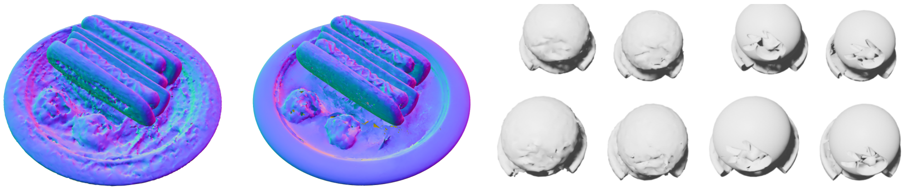
The pipeline of our method is illustrated in Fig. 2. Before training, we employ a volumetric rendering technique to obtain an initial coarse mesh similar to existing methods [9, 39, 45, 57]. During the training stage, we leverage a geometry learner to update both the vertices and the vertex normals (Sec. 3.1). We further learn a shader that decomposes diffuse and specular color (Sec. 3.2), where the specular branch is designed to be able to model highly reflective surfaces by combining view-independent specular features and reflective direction-conditioned environment features. After training with loss functions introduced in Sec. 3.3, we perform UV unwrapping and bake both view-independent and view-dependent features to enable real-time rendering on various devices (Sec. 3.4).

More formally, let $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ denote the initial coarse mesh where $\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^3\}$ is the set of vertices and $\mathcal{F}$ is the set of faces. We further compute initial vertex normals $\mathcal{N} = \{\mathbf{n_v} \in \mathbb{R}^3\}$ given the initial mesh. Let $\mathbf{p} \in \mathbb{R}^3$ denote a rendered surface point on the mesh, $\mathbf{n_p} \in \mathbb{R}^3$ the corresponding normal, $\boldsymbol{\omega}_o \in \mathbb{S}^2$ the viewing direction, and $\boldsymbol{\omega}_r \in \mathbb{S}^2$ the reflective direction. During training, our goal is to refine the mesh $\mathcal{M}$, as well as map $(\mathbf{p}, \boldsymbol{\omega}_r)$ to a color $\mathbf{c} \in \mathbb{R}^3$ to enable mesh-based real-time rendering.

3.1 Geometry Learner

Before training, we follow NeRF2Mesh [45] to extract the initial coarse mesh through grid-based representation [34]. This allows for better modeling of depth discontinuities in geometry [50] and prevents the optimization of mesh geometry from getting stuck in local optima. Then, we propose to leverage a geometry learner to refine the geometry of the initial coarse mesh, including refining the vertex positions and normals.

Vertex Offset: As shown in Fig. 3, the initial mesh extracted from NeRF-based methods may be of low quality. Existing methods [45,54] directly update the vertex positions through gradient backpropagation, which means the vertex update step size is determined by the learning rate and the gradient backpropagated. Thus, costly and difficult manual learning rate tuning is needed to adapt update step sizes for different objects, with a lack of robustness as shown in our experiments. To address this, our method uses a hybrid representation to provide a



(a) Initial normal. **(b)** Optimized normal. **(c)** Initial mesh. **(d)** Optimized mesh.

Fig. 3: Mesh Optimization. The geometry of the initial mesh is often poor. We are able to significantly improve the geometry of the mesh through geometry learner.

learned, adaptive step size for different vertices and different objects:

$$\mathbf{v}' = \mathbf{v} + \Delta\mathbf{v}, \quad \Delta\mathbf{v} = g_{\theta_\mathbf{v}}(\mathbf{v}) \tag{1}$$

where $\mathbf{v}'$ is the updated vertex, $g_{\theta_\mathbf{v}}$ is a multi-resolution hash encoding in combined with a small MLP [34]. Though [47] also uses MLP for vertex offset learning, our coarse-to-fine hybrid representation allows us to leverage local information as well as global information, leading to better modeling of the geometry.

Normal Offset: Our method relies on surface normals to estimate the reflection direction $\boldsymbol{\omega}_r$ to model view-dependent appearance. Following the classical idea of smooth shading, we leverage smoothly interpolated vertex normals to approximate our surface normal. Obtaining accurate vertex normals is yet challenging, especially when the number of faces is limited. Hence, we learn a per-vertex normal offset by taking $\mathbf{v}$ and $\mathbf{n}_\mathbf{v}$ as input:

$$\mathbf{n}'_\mathbf{v} = \mathbf{n}_\mathbf{v} + \Delta\mathbf{n}_\mathbf{v}, \quad \Delta\mathbf{n}_\mathbf{v} = g_{\theta_\mathbf{n}}(\mathbf{v}, \mathbf{n}_\mathbf{v}) \tag{2}$$

where $\mathbf{n}'_\mathbf{v}$ is the learned per-vertex normal and $g_{\theta_\mathbf{n}}$ is another multi-resolution hash encoding-based network. In the experimental section, we demonstrate that learning such a per-vertex normal allows for modeling more accurate surface normals without increasing the number of vertices and faces. Note that $g_{\theta_\mathbf{v}}$ and $g_{\theta_\mathbf{n}}$ are only used during training, as we can directly save the updated vertices $\mathbf{v}'$ and normals $\mathbf{n}'_\mathbf{v}$ for real-time rendering.

3.2 Color Formulation

We decompose the final color $\mathbf{c}$ into a diffuse color $\mathbf{c}_d$ and a specular one $\mathbf{c}_s$:

$$\mathbf{c} = \min(\max(\mathbf{c}_d + \mathbf{c}_s, 0), 1) \tag{3}$$

where $\mathbf{c}_d + \mathbf{c}_s$ is clamped to $[0, 1]$, both individual terms $\mathbf{c}_d$ and $\mathbf{c}_s$ from sigmoid are in [0,1]. We now elaborate on the diffuse and specular color formulation.

Diffuse Color Formulation: With the updated mesh $\mathcal{M}' = (\mathcal{V}', \mathcal{N}', \mathcal{F})$, we perform differentiable rasterization [26] to obtain the position $\mathbf{p}$ corresponding to each pixel. Subsequently, $\mathbf{p}$ is mapped to a diffuse color $\mathbf{c}_d \in \mathbb{R}^3$ and a view-independent feature $\mathbf{f}_s \in \mathbb{R}^3$ which will be used later for decoding specular color:

$$\mathbf{c}_d, \mathbf{f}_s = f_\phi(\mathbf{p}) \tag{4}$$

While the number of channels of $\mathbf{f}_s$ is flexible, we observe that such a compact three-channel representation is sufficient for our purpose.

Specular Color Formulation: Recent non-real-time methods [14, 46, 56] propose to model view-dependent color based on the reflective direction $\boldsymbol{\omega}_r$, instead of the viewing direction used by NeRF. This approach has been shown to effectively model reflective surfaces. Motivated by these methods, we calculate the reflective direction as follows:

$$\boldsymbol{\omega}_r = 2(\boldsymbol{\omega}_o \cdot \mathbf{n_p})\mathbf{n_p} - \boldsymbol{\omega}_o \tag{5}$$

where $\mathbf{n_p}$ is smoothly interpolated surface normal at $\mathbf{p}$ based on our learned vertex normal $\mathbf{n}'_{\mathbf{v}}$. Next, the specular color can be obtained as follows:

$$\mathbf{c}_s = f_\psi(\mathbf{f}_s, \boldsymbol{\omega}_r, (\boldsymbol{\omega}_o \cdot \mathbf{n_p})) \tag{6}$$

where $(\boldsymbol{\omega}_o \cdot \mathbf{n_p})$ is considered as input to model the Fresnel effects [21] and f_ψ needs to be a tiny MLP to enable real-time rendering.

Environment Learner: In practice, we observe that naïvely mapping the reflective viewing direction to specular color (see Eq. (6)) leads to unsatisfying performance due to the capacity limitation of the tiny MLP f_ψ. For a similar reason, Ref-NeRF adopts an 8-layer MLP to map $\mathbf{f}_s$ and $\boldsymbol{\omega}_r$ to the view-dependent color. However, increasing the capacity of f_ψ hampers the real-time rendering capability. Therefore, instead of increasing the capacity of f_ψ, we propose to leverage a neural environment learner of large capacity that maps $\boldsymbol{\omega}_r$ to an environment feature $\mathbf{f}_e \in \mathbb{R}^M$ (M=3 in our paper):

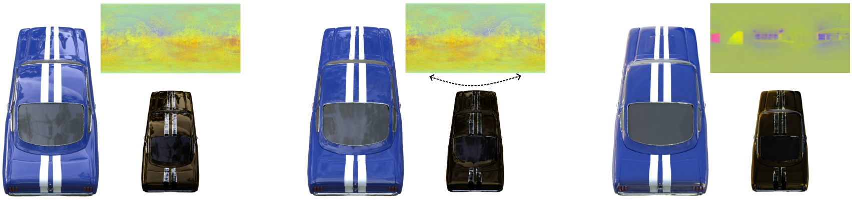
$$\mathbf{f}_e = f_e(\gamma(\boldsymbol{\omega}_r)) \tag{7}$$

where $\gamma(\cdot)$ denotes positional encoding and f_e is a 4-layer MLP. This leads to our final implementation of the specular color:

$$\mathbf{c}_s = f_\psi(\mathbf{f}_s, \mathbf{f}_e, (\boldsymbol{\omega}_o \cdot \mathbf{n_p})) \tag{8}$$

Note that $\mathbf{f}_e$ can be baked into a two-dimensional neural environment feature map for real-time rendering as it solely depends on $\boldsymbol{\omega}_r$.

Scene Editing: REFRAME can decouple geometry, diffuse and specular color, so we can perform simple scene editing tasks. For example, we can edit the geometry and appearance of objects by modifying the mesh or texture image. Additionally, we can perform relighting on the objects by modifying the environment feature map of the scene, as shown in Fig. 4.



(a) Orginal image. (b) Flip lighting. (c) Change lighting.

Fig. 4: Relighting. We can achieve relighting of objects by editing our environment feature map. The image illustrates the editing process of flipping or replacing the environmental feature map. The left side shows the rendering result, the top right corner presents the corresponding environment feature map, and the specular color is displayed in the bottom right corner.

3.3 Loss Functions

We train our model based on image reconstruction loss and several regularization terms. At each iteration, we render a full image and train with the corresponding ground truth. Let $\hat{\mathbf{c}}$ denote the ground truth RGB value at one pixel. We optionally use ground truth binary mask for supervision at synthetic object-centric scenes. Our full loss function is shown in Eq. (9):

$$\begin{aligned}\mathcal{L} =& \lambda_{\mathbf{c}}\mathcal{L}_{\mathbf{c}} + \lambda_{\mathbf{c}_d}\mathcal{L}_{\mathbf{c}_d} + \lambda_{SSIM}\mathcal{L}_{SSIM} + \\ & \lambda_{mask}\mathcal{L}_{mask} + \lambda_{\Delta\mathbf{n_v}}\mathcal{L}_{\Delta\mathbf{n_v}} + \lambda_{max}\mathcal{L}_{max}\end{aligned} \tag{9}$$

Image Reconstruction Loss: Our model is first supervised by the L2 loss between the final rendered color and the ground truth.

$$\mathcal{L}_{\mathbf{c}} = ||\mathbf{c} - \hat{\mathbf{c}}||_2 \tag{10}$$

To encourage the correct decoupling of diffuse and specular colors, we add a diffuse loss to guide the diffuse color to be close to the ground truth color:

$$\mathcal{L}_{\mathbf{c}_d} = ||\mathbf{c}_d - \hat{\mathbf{c}}||_2 \tag{11}$$

We also introduce SSIM [17] loss for better perceptual quality. On object-centric datasets, we additionally utilize mask loss to ensure the rasterization mask closely aligns with the ground truth mask. Note that the mask loss is optional and is not required on unbounded datasets.

Regularizations: We regularize the predicted vertex normal $\mathbf{n}'_\mathbf{v}$ to be close to the original vertex normal $\mathbf{n_v}$ computed from the mesh. This allows for stabilizing the optimization of the vertex normals, *e.g.*, avoid completely flipping the normal. This is achieved by regularizing the magnitude of the normal offset:

$$\mathcal{L}_{\Delta\mathbf{n_v}} = ||\Delta\mathbf{n_v}||_1 \tag{12}$$

We introduce another regularization to encourage the specular color $\mathbf{c}_s$ to be reasonable when visualized alone. As shown in Eq. (3), the final color $\mathbf{c}$ will be clamped when the sum of specular color $\mathbf{c}_s$ and diffuse color $\mathbf{c}_d$ exceeds 1. In this case, the gradient is truncated due to the clamp operation, preventing the backpropagation from updating the color values. This may leave an undesired large specular color $\mathbf{c}_s$ with no penalization. Therefore, we encourage the sum of specular color and diffuse color to not exceed 1:

$$\mathcal{L}_{max} = ||max(\mathbf{c}_d + \mathbf{c}_s - 1, 0)|| \tag{13}$$

3.4 Real-Time Rendering

Environment Feature Map: During the training process, we employ a four-layer MLP with a width of 256 as our environment learner. If we query $\mathbf{f}_e$ through the environment learner module during the inference process, it would significantly slow down the inference speed. Hence, during the inference phase, we bake the learned environment feature $\mathbf{f}_e$ into a 2D environment feature map by converting $\boldsymbol{\omega}_r \in \mathbb{S}^2$ to its polar coordinate, see supplementary for more details. This allows us to simply query the corresponding feature on the environment feature map based on $\boldsymbol{\omega}_r$, thereby greatly accelerating our inference speed, reaching above 200 FPS on a single NVIDIA 3090 GPU.
Texture Images: To further reduce the inference burden and better deploy our method across different platforms, we can also bake the $\mathbf{c}_d$ and $\mathbf{f}_s$ obtained from querying the f_ϕ into two texture images. Specifically, we map the vertices of the mesh to UV coordinates and then bake the features into two texture images, respectively. In order to obtain the normal $\mathbf{n_p}$ from the learned normal vector $\mathbf{n}'_\mathbf{v}$ during real-time rendering, we also bake the normal $\mathbf{n_p}$ into a texture image.

4 Experiment

Implementation Details. Training: Prior to the training stage, we employ the classic quadric error metrics algorithm [13] to simplify the initial mesh to 75,000 faces except for the outdoor scenes which typically have complex geometry. We train for 250 epochs on each scene, with a training time of approximately 2 hours. We utilize the Adam optimizer [23] and employ the cosine annealing learning rate adjustment strategy [32] during the training stage. **Rendering**: The resolution of our baked texture image is 4096×4096, and the resolution of the baked environment feature map is 720×360. As referenced in Sec. 3.4, once we have baked the environment learner into the environment feature map, our approach is already capable of achieving above 200 FPS on the GPU (referred to as Ours). This implementation ensures a fair comparison with the baselines since their results quoted are evaluated using GPU implementation that can not directly deploy on mobile devices. Given that we use texture images for rendering when implemented on mobile devices, we also employ texture images on the GPU for quality evaluation (referred to as Ours (Mobile)). Following NeRF2Mesh [45] and MobileNeRF [9], we employ anti-aliasing to enhance our rendering quality. For details on the implementation of anti-aliasing, please refer to the supplementary material. All experiments are performed on a single NVIDIA 3090 GPU.

	NeRF Synthetic Dataset			Shiny Blender Dataset			Real Captured Dataset		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [33]	31.01	0.947	0.081	-	-	-	-	-	-
Ref-NeRF [46]	33.99	0.966	0.038	35.96	0.967	0.058	24.45	0.665	0.142
3DGS [22]	33.30	0.969	0.030	30.37	0.947	0.083	24.06	0.661	0.259
NvDiffRec [35]	29.05	0.939	0.081	29.05	0.938	0.111	-	-	-
MobileNeRF [9]	30.90	0.947	0.062	26.62	0.883	0.163	22.96	0.493	0.430
NeRF2Mesh [45]	29.67	0.940	0.072	26.96	0.896	0.170	22.71	0.523	0.419
Ours	31.04	0.961	0.040	36.02	0.983	0.040	22.50	0.598	0.325
Ours (Mobile)	30.84	0.959	0.042	35.83	0.981	0.041	22.11	0.524	0.423

Table 1: Rendering Quality. Baseline comparisons of the rendering quality on three different datasets. Red represents the optimal, orange represents the second best, and yellow represents the third.

Dataset. We validate the effectiveness and robustness of our method using three different datasets with varied scenes. 1) **NeRF [33] Synthetic Dataset**: This dataset consists of eight synthetic scenes. 2) **Shiny Blender Dataset**: This dataset is introduced by Ref-NeRF [46] and includes six glossy objects, serving as the primary dataset for assessing the reflective surface modeling capabilities of the methods. 3) **Real Captured Dataset**: We select three outdoor scenes with rich reflective appearances same as Ref-NeRF [46] to validate the effectiveness of our method in real-world outdoor capture scenarios. Due to the poor quality of the initial mesh extracted by Nerf2Mesh [45] when modeling glossy objects, we utilize the meshes extracted by Neuralangelo [29] and Ref-NeuS [14] as our initialization in certain scenes, see supplementary for more details.
Baseline. We compare with various types of baselines, including Ref-NeRF [46], 3DGS [22], NvDiffRec [35], MobileNeRF [9], and NeRF2Mesh [45] in our study. Ref-NeRF [46] serves as the state-of-the-art volume rendering method for handling the strong reflective appearance. 3DGS [22] represents a powerful new representation capable of real-time rendering on GPU. NvDiffRec [35] represents physically based rendering methods. MobileNeRF [9], and NeRF2Mesh [45] are representative works in the field of real-time rendering for mobile devices. We import the results of the baselines from their original paper and replicate the results that missing in their original paper.

4.1 Reconstruction Quality

Rendering Quality. We compare the rendering quality of our method and baseline methods on three datasets, see Tab. 1. It is worth noting that Ref-NeRF [46] is the state-of-the-art method for modeling glossy objects but cannot achieve real-time rendering, even on high-end GPU. We introduce Ref-NeRF [46] as a benchmark for rendering quality and demonstrate that our method achieves comparable or even superior rendering quality to non-real-time state-of-the-art methods. While 3DGS [22] can render in real-time on GPU and has higher PSNR on both NeRF Synthetic and Real Captured dataset than ours, it's important

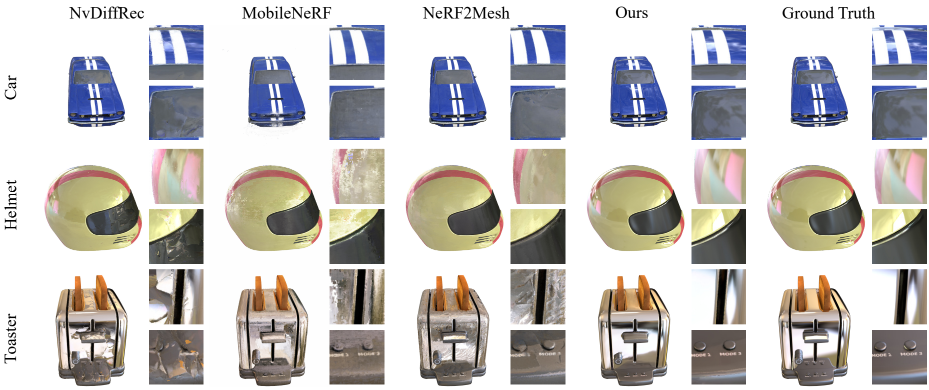


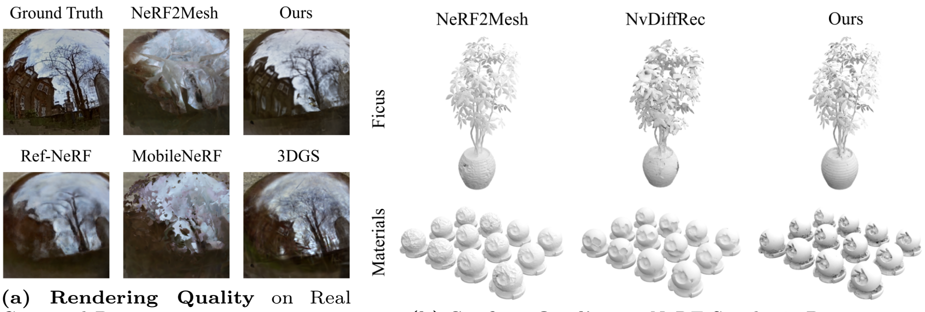
Fig. 5: Rendering Quality on Shiny Blender Dataset. Our method achieves optimal rendering quality in most scenes and provides better modeling of reflective appearance compared to the comparison methods [9, 35, 45].

to note that 3DGS struggles to model reflective surfaces effectively (Fig. 6a). Moreover, they have a larger memory overhead than us. *E.g.*, for gardenspheres, the memory cost for 3DGS is 1.4GB, whereas our method only requires 84.3MB.

In comparison to mesh-based real-time rendering methods [9, 35, 45], we achieve optimal rendering quality in most object-centric datasets, as shown in Fig. 5. Despite having mask supervision on object-centric datasets as well, NvDiffRec [35] and NeRF2Mesh [45] struggle to achieve high rendering quality. This could be due to the fact that NvDiffRec relies on the simplified rendering equation, whereas NeRF2Mesh's simple color formulation struggles to model highly reflective objects. Note that NeRF2Mesh also requires an initial mesh in its second stage, and we provide NeRF2Mesh with the same initial mesh as ours for all experiments for a fair comparison.

In unbounded outdoor scenes without masks, our method still achieves high-quality rendering, as demonstrated in Fig. 6a. Although our PSNR may be slightly lower compared to the baselines, we outperform them in rendering foreground glossy objects. However, this advantage is not accurately reflected in the PSNR metric since the foreground reflective objects occupy only a small region in the image. Further, our method mainly exhibits artifacts in the background, this is due to the fact that the initial mesh can only faithfully reconstruct the foreground but not the background, see supplementary for more details.

Surface Quality. Furthermore, we observe that our method not only improves the rendering quality when modeling glossy objects but also generates more accurate meshes, as shown in Fig. 6b. It can be seen that the meshes obtained from [35, 45] often have poor quality in the reflective region while our method is capable of accurately reconstructing the geometry.



(a) **Rendering Quality** on Real Captured Dataset.

(b) **Surface Quality** on NeRF Synthetic Dataset.

Fig. 6: Qualitative Comparison. Compared to the baselines, our method reconstructs reflective regions of higher fidelity and yields more accurate surface geometry.

	MacBook Pro	iPhone12	iPad Air3	Legion Y-7000	HUAWEI nova7
MobileNeRF [9]	120.00	60.00	60.00	55.31	39.19
NeRF2Mesh [45]	120.00	60.00	60.00	48.69	32.56
Ours (Mobile)	120.00	60.00	60.00	48.50	31.71

Table 2: Computational Efficiency Comparison. We test the FPS of our method and the baseline methods [9, 45] on the NeRF Synthetic dataset across multiple devices. On MacBook Pro, iPhone 12, and iPad Air3, both our method and the baseline methods [9,45] achieve the maximum screen refresh rate.

4.2 Rendering Efficiency

Computational efficiency. During the mobile rendering stage, the bottleneck of our rendering speed lies in querying the tiny MLP f_ψ to obtain the specular color. However, due to our MLP being only two layers with 64 widths, our rendering speed remains relatively fast. Therefore, our method achieves real-time rendering on both consumer GPUs and mobile devices. We compare the rendering speed with the baselines [9, 45] that can render in real-time on mobile devices across multiple platform devices, as seen in Tab. 2.
Memory efficiency. Our cache mainly consists of texture images, the environment feature map, and the mesh. Due to our high-quality estimation of vertex normals, our mesh can achieve high-quality rendering with a low number of vertices and faces, leading to a small cache size in most scenes, as shown in Tab. 3.

4.3 Ablation Study

Geometry Learner. [54] and [45] utilize direct gradient backpropagation to update the vertex positions of the mesh. We employ this update strategy in our method and verify that this update method lacks robustness. Directly using the backpropagated gradients to update the vertex positions can lead to a devastating blow to the mesh, especially when using a large learning rate. This

	NeRF Synthetic Dataset			Shiny Blender Dataset		
	#V (10^3)	#F (10^3)	Cache	#V (10^3)	#F (10^3)	Cache
MobileNeRF [9]	494	224	125.75	1028	343	152.37
NeRF2Mesh [45]	200	192	73.53	45	91	22.15
Ours (Mobile)	37	75	47.55	38	75	52.86

Table 3: Memory Efficiency. While exhibiting higher quality, our method maintains a relatively low memory consumption compared to baselines. Cache reported in MB.

	w/o $g_{\theta_\mathbf{v}}$		Ours
	L=0.001	L=0.0001	L=0.001
Toaster	17.01	17.01	24.95
Materials	23.01	29.06	29.58

Table 4: Ablation of Vertex Offset Learning. L represents the learning rate. Reported in PSNR.

		w/o $\Delta\mathbf{n}_\mathbf{v}$	Ours
Toaster	#F=10k	24.24	24.62
	#F=75k	24.55	24.95
Materials	#F=10k	27.69	28.23
	#F=75k	29.46	29.58

Table 5: Ablation of Normal Offset Learning. Reported in PSNR.

significantly compromises the rendering quality, as shown in Tab. 4. Additionally, different datasets require different learning rates, as indicated in Tab. 4. While a learning rate of 0.0001 works fine for the materials dataset, it performs poorly on the toaster dataset. In contrast, our vertex update strategy not only exhibits robustness across different datasets but also provides higher rendering quality.

In order to verify the necessity of estimating an offset for the vertices normal, we conduct experiments with the w/o normal offset setting. In this setting, we do not estimate a normal offset for each vertex, but we still compute an updated normal based on the updated vertex positions. As shown in Tab. 5, introducing per-vertex normal offset provides a gain in reconstruction quality. This gain is particularly significant when the number of faces is low. Notably, in the case of the toaster dataset, a mesh with 10k faces with normal offset even achieves rendering quality surpassing that of a mesh with 75k faces without normal offset.

Environment Learner. To validate the effectiveness of the environment learner f_e, we directly input $\boldsymbol{\omega}_r$ values into the tiny MLP f_ψ as shown in Eq. (6), similar to [9] and [45], except they use viewing direction instead of reflective direction. Experimental results demonstrate that directly inputting $\boldsymbol{\omega}_r$ into the tiny MLP f_ψ cannot learn the correct reflection appearance as a result of the limited network's capacity, as depicted in Fig. 7. Therefore, introducing the environment learner module improves the reconstruction quality without compromising the inference speed, as demonstrated in Tab. 6.

Environment Feature Map: We can directly optimize an environment feature map instead of learning it through the environment learner module and baking it. However, as shown in Tab. 7, the direct optimization of the environment feature map performs poorly when the map resolution is high. This is because, at higher resolutions, neighboring directions are mapped to distant grid points, leading to a non-smooth interpolation. As a result, the grid points can-

(a) w/o f_e **(b)** Ours **(c)** Ground Truth

Fig. 7: Ablation of Environment Learner. Eliminating f_e results in bad rendering quality.

	w/o f_e	Ours
Toaster	19.31	24.95
Materials	27.37	29.58

Table 6: Ablation of Environment Learner. Reported in PSNR.

	Resolutions H*W	Ours PSNR	Ours Cache(KB)	Directly optimization PSNR	Directly optimization Cache(KB)
Toaster	180*360	24.86	49	24.16	91
	900*1800	24.96	504	23.59	2460
Materials	180*360	29.55	63	29.58	114
	900*1800	29.58	706	29.03	3000

Table 7: Ablation of Environment Feature Map. Our method outperforms in both cache size and rendering quality.

not share global information and only contain its own local information. On the other hand, when the resolution is too low, the rendering quality is limited by the capacity of the environment feature map. In contrast, our method exhibits good rendering quality across different resolutions by learning the environment feature map via an MLP. Additionally, our environment feature maps enjoy the smoothness bias of the MLP, resulting in relatively smooth tensor maps. Consequently, when saving images in PNG format, our method demonstrates good compression performance. This further reduces the overhead of our caching.

5 Conclusion

Although our method successfully models the appearance of glossy objects, similar to Ref-NeRF [46], it still struggles with accurately modeling interreflections and non-distant illumination. Additionally, our method requires a certain level of quality in the initial mesh.

In summary, our method achieves real-time rendering on edge devices with low hardware budgets. Besides, we propose a novel approach for modeling view-dependent appearance and can optimize the appearance and geometry of glossy objects with high computational efficiency and low memory footprint. Furthermore, by effectively decoupling the scene's geometry, appearance, and environmental information, our method can perform simple scene editing tasks.

Acknowledgements

This work is supported by NSFC under grant 62202418 and U21B2004. We thank Xiaoran Cao, Chenxi Tu, and Sicheng Li for their valuable discussions.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. arXiv preprint arXiv:2304.06706 (2023)
4. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition. arXiv preprint arXiv:2008.03824 (2020)
5. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12684–12694 (2021)
6. Boss, M., Jampani, V., Braun, R., Liu, C., Barron, J., Lensch, H.: Neural-pil: Neural pre-integrated lighting for reflectance decomposition. Advances in Neural Information Processing Systems **34**, 10691–10704 (2021)
7. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient geometry-aware 3d generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16123–16133 (2022)
8. Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5799–5809 (2021)
9. Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16569–16578 (2023)
10. Esposito, S., Baieri, D., Zellmann, S., Hinkenjann, A., Rodola, E.: Kiloneus: A versatile neural implicit surface representation for real-time rendering. arXiv preprint arXiv:2206.10885 (2022)
11. Fu, X., Zhang, S., Chen, T., Lu, Y., Zhu, L., Zhou, X., Geiger, A., Liao, Y.: Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. In: 2022 International Conference on 3D Vision (3DV). pp. 1–11. IEEE (2022)
12. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14346–14355 (2021)
13. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques. pp. 209–216 (1997)
14. Ge, W., Hu, T., Zhao, H., Liu, S., Chen, Y.C.: Ref-neus: Ambiguity-reduced neural implicit surface learning for multi-view reconstruction with reflection. arXiv preprint arXiv:2303.10840 (2023)

15. Guo, Y.C., Cao, Y.P., Wang, C., He, Y., Shan, Y., Zhang, S.H.: Vmesh: Hybrid volume-mesh representation for efficient view synthesis. In: SIGGRAPH Asia 2023 Conference Papers. pp. 1–11 (2023)
16. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5875–5884 (2021)
17. Hore, A., Ziou, D.: Image quality metrics: Psnr vs. ssim. In: 2010 20th international conference on pattern recognition. pp. 2366–2369. IEEE (2010)
18. Hu, T., Liu, S., Chen, Y., Shen, T., Jia, J.: Efficientnerf efficient neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12902–12911 (2022)
19. Jin, H., Liu, I., Xu, P., Zhang, X., Han, S., Bi, S., Zhou, X., Xu, Z., Su, H.: Tensoir: Tensorial inverse rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2023)
20. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques. pp. 143–150 (1986)
21. Kautz, J., McCool, M.D.: Approximation of glossy reflection with prefiltered environment maps. In: Graphics Interface. vol. 2000, pp. 119–126 (2000)
22. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
23. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
24. Kundu, A., Genova, K., Yin, X., Fathi, A., Pantofaru, C., Guibas, L.J., Tagliasacchi, A., Dellaert, F., Funkhouser, T.: Panoptic neural fields: A semantic object-aware neural scene representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12871–12881 (2022)
25. Kurz, A., Neff, T., Lv, Z., Zollhöfer, M., Steinberger, M.: Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII. pp. 254–270. Springer (2022)
26. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics (TOG) **39**(6), 1–14 (2020)
27. Li, C., Li, S., Zhao, Y., Zhu, W., Lin, Y.: Rt-nerf: Real-time on-device neural radiance fields towards immersive ar/vr rendering. In: Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design. pp. 1–9 (2022)
28. Li, S., Li, H., Wang, Y., Liao, Y., Yu, L.: Steernerf: Accelerating nerf rendering via smooth viewpoint trajectory. arXiv preprint arXiv:2212.08476 (2022)
29. Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8456–8465 (2023)
30. Liang, R., Chen, H., Li, C., Chen, F., Panneer, S., Vijaykumar, N.: Envidr: Implicit differentiable renderer with neural environment lighting. arXiv preprint arXiv:2303.13022 (2023)
31. Lin, H., Peng, S., Xu, Z., Yan, Y., Shuai, Q., Bao, H., Zhou, X.: Efficient neural radiance fields for interactive free-viewpoint video. In: SIGGRAPH Asia 2022 Conference Papers. pp. 1–9 (2022)
32. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)

33. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
34. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
35. Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting triangular 3d models, materials, and lighting from images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8280–8290 (2022)
36. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
37. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021)
38. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10318–10327 (2021)
39. Rakotosaona, M.J., Manhardt, F., Arroyo, D.M., Niemeyer, M., Kundu, A., Tombari, F.: Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. arXiv preprint arXiv:2303.09431 (2023)
40. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14335–14345 (2021)
41. Rematas, K., Liu, A., Srinivasan, P.P., Barron, J.T., Tagliasacchi, A., Funkhouser, T., Ferrari, V.: Urban radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12932–12942 (2022)
42. Schwarz, K., Liao, Y., Niemeyer, M., Geiger, A.: Graf: Generative radiance fields for 3d-aware image synthesis. Advances in Neural Information Processing Systems **33**, 20154–20166 (2020)
43. Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7495–7504 (June 2021)
44. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8248–8258 (2022)
45. Tang, J., Zhou, H., Chen, X., Hu, T., Ding, E., Wang, J., Zeng, G.: Delicate textured mesh recovery from nerf via adaptive surface refinement. arXiv preprint arXiv:2303.02091 (2023)
46. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5481–5490. IEEE (2022)
47. Walker, T., Mariotti, O., Vaxman, A., Bilen, H.: Explicit neural surfaces: Learning continuous geometry with deformation fields. arXiv preprint arXiv:2306.02956 (2023)

48. Wang, H., Ren, J., Huang, Z., Olszewski, K., Chai, M., Fu, Y., Tulyakov, S.: R2l: Distilling neural radiance field to neural light field for efficient novel view synthesis. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI. pp. 612–629. Springer (2022)
49. Wang, L., Zhang, J., Liu, X., Zhao, F., Zhang, Y., Zhang, Y., Wu, M., Yu, J., Xu, L.: Fourier plenoctrees for dynamic radiance field rendering in real-time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13524–13534 (2022)
50. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
51. Wang, Z., Li, L., Shen, Z., Shen, L., Bo, L.: 4k-nerf: High fidelity neural radiance fields at ultra high resolutions. arXiv preprint arXiv:2212.04701 (2022)
52. Wang, Z., Shen, T., Gao, J., Huang, S., Munkberg, J., Hasselgren, J., Gojcic, Z., Chen, W., Fidler, S.: Neural fields meet explicit geometric representations for inverse rendering of urban scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8370–8380 (2023)
53. Weng, C.Y., Curless, B., Srinivasan, P.P., Barron, J.T., Kemelmacher-Shlizerman, I.: Humannerf: Free-viewpoint rendering of moving people from monocular video. In: Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition. pp. 16210–16220 (2022)
54. Worchel, M., Diaz, R., Hu, W., Schreer, O., Feldmann, I., Eisert, P.: Multi-view mesh reconstruction with neural deferred shading. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6187–6197 (2022)
55. Wu, L., Lee, J.Y., Bhattad, A., Wang, Y.X., Forsyth, D.: Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16200–16209 (2022)
56. Wu, T., Sun, J.M., Lai, Y.K., Gao, L.: De-nerf: Decoupled neural radiance fields for view-consistent appearance editing and high-frequency environmental relighting. In: ACM SIGGRAPH 2023 Conference Proceedings. pp. 1–11 (2023)
57. Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: Bakedsdf: Meshing neural sdfs for real-time view synthesis. arXiv preprint arXiv:2302.14859 (2023)
58. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5752–5761 (2021)
59. Zhang, J., Yang, G., Tulsiani, S., Ramanan, D.: Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. Advances in Neural Information Processing Systems **34**, 29835–29847 (2021)
60. Zhang, J., Yao, Y., Li, S., Liu, J., Fang, T., McKinnon, D., Tsin, Y., Quan, L.: Neilf++: Inter-reflectable light fields for geometry and material estimation. arXiv preprint arXiv:2303.17147 (2023)
61. Zhang, K., Luan, F., Li, Z., Snavely, N.: Iron: Inverse rendering by optimizing neural sdfs and materials from photometric images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5565–5574 (2022)
62. Zhang, K., Luan, F., Wang, Q., Bala, K., Snavely, N.: Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5453–5462 (2021)

63. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. ACM Transactions on Graphics (ToG) **40**(6), 1–18 (2021)
64. Zhang, Y., Sun, J., He, X., Fu, H., Jia, R., Zhou, X.: Modeling indirect illumination for inverse rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 18643–18652 (June 2022)
65. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.J.: In-place scene labelling and understanding with implicit scene representation. In: ICCV (2021)