Delicate Textured Mesh Recovery from NeRF via Adaptive Surface Refinement

Jiaxiang Tang[1], Hang Zhou[2], Xiaokang Chen[1], Tianshu Hu[2], Errui Ding[2], Jingdong Wang[2], Gang Zeng[1]
[1]School of Intelligence Science and Technology, Peking University [2]Baidu Inc.
{tjx, pkucxk, zeng}@pku.edu.cn {zhouhang09,hutianshu01,dingerrui,wangjingdong}@baidu.com

https://me.kiui.moe/nerf2mesh

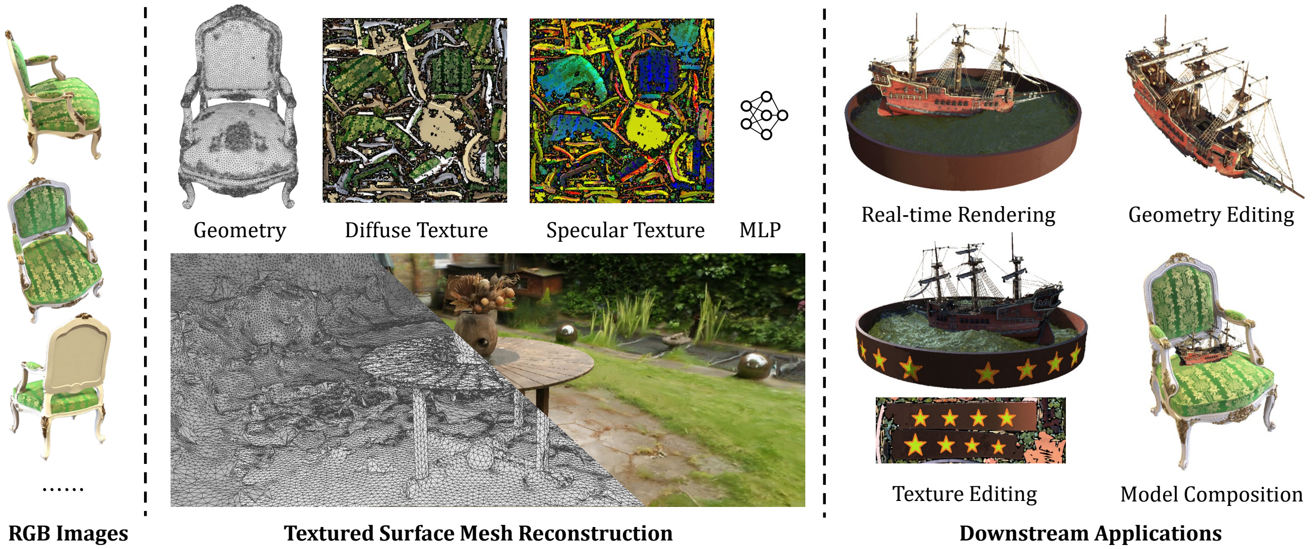

Figure 1: Our framework, *NeRF2Mesh*, reconstructs high-quality surface meshes with diffuse and specular textures from multi-view RGB images, generalizing well from object- to scene-level datasets. The exported textured meshes are ready-to-use for common graphics hardware and software, facilitating various downstream applications.

Abstract

Neural Radiance Fields (NeRF) have constituted a remarkable breakthrough in image-based 3D reconstruction. However, their implicit volumetric representations differ significantly from the widely-adopted polygonal meshes and lack support from common 3D software and hardware, making their rendering and manipulation inefficient. To overcome this limitation, we present a novel framework that generates textured surface meshes from images. Our approach begins by efficiently initializing the geometry and view-dependency decomposed appearance with a NeRF. Subsequently, a coarse mesh is extracted, and an iterative surface refining algorithm is developed to adaptively adjust both vertex positions and face density based on re-projected rendering errors. We jointly refine the appearance with geometry and bake it into texture images for real-time rendering. Extensive experiments demonstrate that our method achieves superior mesh quality and competitive rendering quality.

1. Introduction

The reconstruction of 3D scenes from RGB images is a complex task in computer vision with many real-world applications. In recent years, Neural Radiance Fields (NeRF) [30, 2, 8, 31] have gained popularity for its impressive ability to reconstruct and render large-scale scenes with realistic details. However, NeRF representations often use implicit functions and specialized ray marching algorithms for rendering, making them difficult to manipulate and slow to render due to poor hardware support, which limits their use in downstream applications. In contrast, polygonal meshes are the most commonly used representation in 3D applications, and are well-supported by most graphic hardware to accelerate rendering. However, direct reconstruction of meshes can be challenging due to their irregularity, and most approaches are limited to fixed topology or object-level reconstructions.

Some recent works [32, 6, 11, 51] have focused on combining the advantages of both NeRF and mesh representation. MobileNeRF [11] proposes to optimize NeRF on a grid mesh and binarize rendering weights to incorporate rasterization for real-time rendering. However, the resulting mesh is distinct from the surface mesh, and the textures are in the feature space instead of the RGB space, making it difficult to edit or manipulate. To obtain accurate surface meshes, a popular approach is to use Signed Distance Fields (SDF), which can accurately define a surface [46, 50, 54]. However, this line of research typically generates over-smoothed geometry that fails to model thin structures, and often ignores the rendering quality. Additionally, meshes obtained through Marching Cubes [28] may have inaccurate vertex positions and a large number of faces. NVdiffrec [32] uses a differentiable rasterizer [22] to optimize a deformable tetrahedral grid, but is limited to object-level reconstruction and also fails to recover complex topology. The presence of a representation gap makes it challenging to recover accurate surface meshes from volumetric NeRF while maintaining rendering quality.

This paper presents a new framework called *NeRF2Mesh* for extracting delicate textured surface meshes from RGB images, as illustrated in Figure 1. Our key insight is to **refine a coarse mesh extracted from NeRF for joint optimization of geometry and appearance**. The volumetric NeRF representation is suitable for efficient initialization of geometry and appearance. With a coarse mesh extracted from NeRF, we adjust the vertices position and face density based on 2D rendering errors, which in turn contributes to appearance optimization. To enable texture editing, we decompose the appearance into view-independent diffuse and view-dependent specular terms, so the diffuse color can be exported as a standard RGB image texture. The specular color is exported as a feature texture that can produce view-dependent color by being fed into a small MLP embedded in the fragment shader along with the current viewing directions. Overall, our framework enables the creation of versatile and practical mesh assets that can be used in a range of scenarios that are challenging for volumetric NeRF.

Our contributions can be summarized as follows:

- We present the NeRF2Mesh framework to reconstruct textured surface meshes from multi-view RGB images, by jointly refining the geometry and appearance of coarse meshes extracted from an appearance decomposed NeRF.
- We propose an adaptive mesh refinement algorithm that enables us to adjust face density, where complex surfaces are subdivided and simpler surfaces are decimated based on re-projected 2D image errors.
- Our method achieves enhanced surface mesh quality, relatively smaller mesh size, and competitive rendering quality to recent methods. Furthermore, the resulting meshes can be real-time rendered and interactively edited with common 3D hardware and software.

2. Related Work

2.1. NeRF for Scene Reconstruction

NeRF [30] and its subsequent works [2, 3, 56, 36, 48, 34, 48, 1, 7, 37] represent a remarkable advancement in 3D scene reconstruction from RGB images. Despite the superior rendering quality, vanilla NeRF faces several issues. For instance, the model's training and inference speed is slow due to the large number of MLP evaluations, which limits the widespread adoption of NeRF representation. To address this, several works [53, 38, 41, 31, 8] have proposed methods to reduce the MLP's size or eliminate it altogether, and instead optimize an explicit 3D feature grid that stores the density and appearance information. DVGO [41] employs two dense feature grids for density and appearance encoding, but the dense grid leads to a large model size. To effectively control the model size, Instant-NGP [31] proposes a multi-resolution hash table. In addition to the efficiency issue, the implicit representation of NeRF cannot be directly manipulated and edited in both geometry and appearance, unlike explicit representations such as polygonal meshes. Although some works [23, 42, 49, 26, 45] explore geometry manipulation and composition of NeRF, they are still limited in different ways. On the other hand, others [57, 5, 55, 40, 4, 44] aim to decompose the reflectance under unknown illumination to enable relighting and texture editing. These problems result in a gap between NeRF representation and widely used polygonal meshes in downstream applications. Our objective is to narrow this gap by exploring methods to convert NeRF reconstructions into textured meshes.

2.2. Surface Mesh for Scene Reconstruction

Reconstructing explicit surface meshes directly can be challenging, particularly for complex scenes with intricate topology. Most approaches in this area of research assume a template mesh with a fixed topology [9, 10, 20, 25]. Recent methods [32, 17, 24, 39] have begun to address topology optimization. NVdiffrec [32] combines differentiable marching tetrahedrons [24] with differentiable rendering to optimize surface meshes directly. It can also decompose materials and illumination, which is further improved in NVdiffrecMC [17] using Monte Carlo rendering. Nonetheless, these methods still have limitations in that they only apply to object-level mesh reconstruction and struggle to differentiate between background and foreground meshes in unbounded outdoor scenes. A foreground mask [32] must be prepared to optimize the object boundary using differentiable rendering. In contrast, our focus is on surface mesh reconstruction at both the object and scene levels without prior knowledge.

2.3. Extracting Surface Mesh from NeRF

NeRF represents geometry using a volumetric density field, which may not necessarily form a concrete surface. To address this, a popular strategy is to learn a Signed Distance Field (SDF) [46, 50, 14, 15, 54, 47], where the surface can be determined by the zero level set. NeuS [46] applies an SDF to density transformation to enable differentiable rendering, and the Marching Cubes [28] algorithm is usually used to extract the surface mesh from these volumes. A concurrent work BakedSDF [51] optimize a hybrid SDF volume-surface representation and bake it into meshes for real-time rendering. However, SDF-based methods tend to learn over-smoothed geometry and fail to handle thin structures. Some methods [43, 27] explore Unsigned Distance Field (UDF) or a combination of density field and SDF to address this limitation, but they are still limited to object-level reconstruction. SAMURAI [6] aims to jointly recover camera poses, geometry, and appearance of a single object under unknown captured conditions and export textured meshes. MobileNeRF [11] proposes to train NeRF on a grid mesh, which can be rendered in real-time. However, their mesh is not exactly the surface mesh and only exports features as texture, which have to be rendered with a custom shader and are unfriendly for editing. Recent works [31, 35] have found that an exponential density activation can help to concentrate the density and form better surfaces. We also adopt density field to capture complex topology and further refine the surface.

3. Method

In this section, we introduce our framework, as shown in Figure 2, for reconstructing a textured surface mesh from a collection of RGB images that is compatible with common 3D hardware and software. The training process comprises two stages. Firstly, we train a grid-based NeRF [31] to efficiently initialize the geometry and appearance of the mesh (Section 3.1). Next, we extract a coarse surface mesh and fine-tune both surface geometry and appearance (Section 3.2). Once the training is complete, we can export a textured surface mesh in standard formats such as wavefront OBJ and PNG, which is ready-to-use for various downstream applications (Section 3.3).

3.1. Efficient NeRF Training (Stage 1)

In the initial stage, we leverage the volumetric NeRF representation to recover both the geometry and appearance of arbitrary scenes. The primary goal of this stage is to **efficiently establish topologically accurate geometry and decomposed appearance in preparation for the subsequent surface mesh refining phase**. While direct work on polygonal meshes [32] presents challenges in learning complex geometries, volumetric NeRF [30] provides a more accessible alternative.

We follow recent advancements in grid-based NeRF [31, 41, 8, 38] to enhance the efficiency of NeRF by employing two separate feature grids to represent the 3D space. Though the surface may lack precision as the density may scatter throughout the space, these issues can be addressed through mesh refinement in the next stage.

Geometry. Geometry learning is facilitated through a density grid [31] and a shallow MLP expressed as follows:

$$\sigma = \phi(\mathrm{MLP}(E^{\text{geo}}(\mathbf{x}))), \tag{1}$$

where ϕ is the exponential activation [31] that promotes sharper surface, E^{geo} is a learnable multi-resolutional feature grid, and $\mathbf{x} \in \mathbb{R}^3$ is the position of any 3D point.

Appearance Decomposition. NeRF typically operates under no assumption of illumination or material properties. As such, previous works have mainly employed a 5D implicit function conditioned on 3D position and 2D view direction to model view-dependent appearance. Despite achieving photo-realistic performance, this approach renders the appearance as a black box, making it challenging to represent with traditional 2D texture images.

To address this issue, we decompose the appearance into view-independent diffuse color $\mathbf{c}_d$ and view-dependent specular color $\mathbf{c}_s$ using a color grid and two shallow MLPs, expressed as follows:

$$\mathbf{c}_d, \mathbf{f}_s = \psi(\mathrm{MLP}_1(E^{\text{app}}(\mathbf{x}))), \tag{2}$$

$$\mathbf{c}_s = \psi(\mathrm{MLP}_2(\mathbf{f}_s, \mathbf{d})), \tag{3}$$

where ψ refers to the sigmoid activation, $\mathbf{f}_s$ represents the intermediate features for the specular color at position $\mathbf{x}$, and $\mathbf{d}$ represents the view direction. The final color is obtained by summing the two terms:

$$\mathbf{c} = \mathbf{c}_d + \mathbf{c}_s, \tag{4}$$

As shown in Figure 3, we successfully separate the diffuse and specular terms. The diffuse color in $\mathbb{R}^3$ can be conveniently baked as an RGB image texture. Meanwhile, the specular features $\mathbf{f}_s$ can also be baked as textures, and the small MLP_2 can be fit into a fragment shader following [11]. Consequently, the specular color can also be exported and rendered later (see Section 3.3 for details). It is important to note that our approach involves baking the lighting conditions into the textures. This is because estimating the environment lighting can be challenging for realistic datasets, and previous studies have observed that this can result in reduced rendering quality [57, 32].

Loss function. To optimize our model, we follow the original NeRF's rendering loss. Given a ray $\mathbf{r}$ originating from $\mathbf{o}$ with direction $\mathbf{d}$, we query the model at positions $\mathbf{x}_i = \mathbf{o} + t_i\mathbf{d}$, sequentially sampled along the ray, for densities σ_i and colors $\mathbf{c}_i$. The final pixel color is obtained by

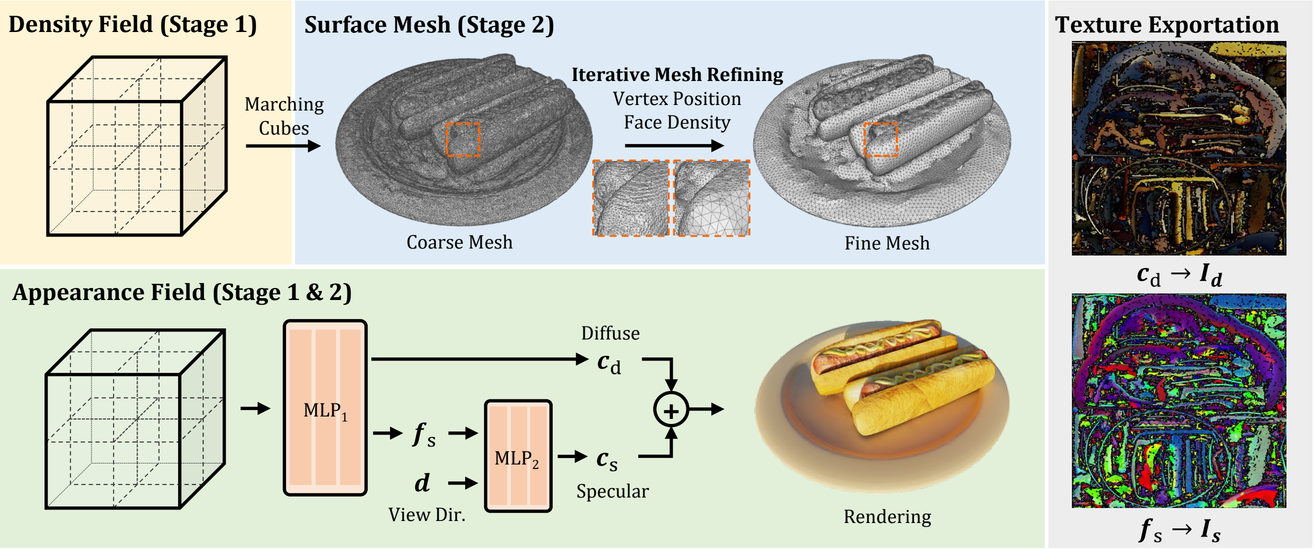

Figure 2: *NeRF2Mesh Framework*. The geometry is initially learned with a density grid, which is then extracted to form a coarse mesh. We optimize it into a fine mesh with more accurate surface and adaptive face density. The appearance is learned with a color grid, and decomposed into diffuse and specular terms. After convergence, we can export the fine mesh, unwrap its UV coordinates, and bake the textures.

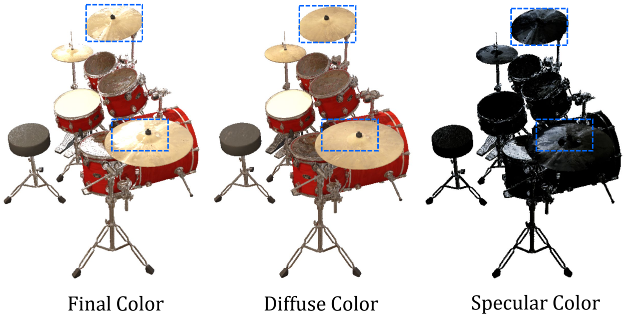

Figure 3: Separation of diffuse color and specular color.

numerical quadrature using the following equation:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_i T_i w_i \mathbf{c}_i, T_i = \prod_{j<i}(1 - w_i), \qquad (5)$$

where $\delta_i = t_{i+1} - t_i$ is the step size, $w_i = 1 - \exp(-\sigma_i \delta_i)$ is the point-wise rendering weight, and T_i is the transmittance. We minimize the loss between each pixel's predicted color $\hat{\mathbf{C}}(\mathbf{r})$ and the ground truth color $\mathbf{C}(\mathbf{r})$:

$$\mathcal{L}_{\text{NeRF}} = \sum_{\mathbf{r}} ||\mathbf{C}(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r})||^2, \qquad (6)$$

We encourage separation of the diffuse and specular terms by applying a L1 regularization on the specular color:

$$\mathcal{L}_{\text{specular}} = \sum_i |\mathbf{c}_s(\mathbf{x}_i)|, \qquad (7)$$

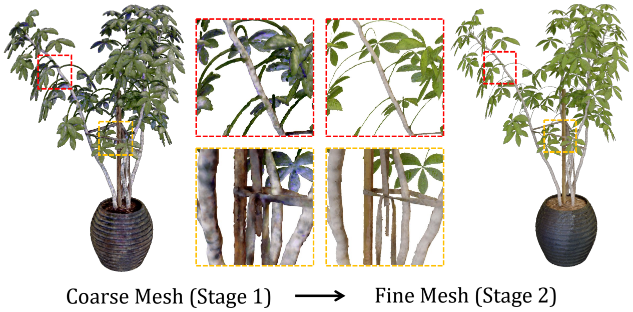

Figure 4: **Mesh Refining**. We refine both the geometry and appearance of the coarse mesh in stage 2.

To make the surface sharper, we apply an entropy regularization on the rendering weights:

$$\mathcal{L}_{\text{entropy}} = -\sum_i (w_i \log w_i + (1 - w_i) \log(1 - w_i)) \qquad (8)$$

where w_i is the per-point rendering weight. For unbounded outdoor scenes, we also apply Total Variation (TV) regularization on the density field E^{geo} to reduce floaters [41, 8].

3.2. Surface Mesh Refining (Stage 2)

In the second stage, we extract a coarse surface mesh from the stage 1 NeRF model and further optimize it. This process involves refining the vertices, triangles, and appearance on the surface, as illustrated in Figure 4.

Appearance refining. To render an image, the mesh undergoes rasterization and the 3D positions are interpolated onto the image space pixel-wisely. Since the pixel colors are still queried in a point-wise manner, the appearance models from stage 1 can be inherited into stage 2. This eliminates the need to learn the appearance from scratch, reducing the required training steps for stage 2 to converge. The pixel-wise color loss in Equation 6 is still applied in stage 2 to allow joint optimization of appearance and geometry.

Iterative mesh refining. The coarse meshes extracted by Marching Cubes [28] from density field are often flawed. These flaws include inaccurate vertices and dense, evenly-distributed faces, leading to vast disk storage and slow rendering speed. Our goal is to recover delicate meshes resembling human-made ones by refining both vertex positions and face density.

Given an initial coarse mesh $\mathcal{M}_{\text{coarse}} = \{\mathcal{V}, \mathcal{F}\}$, we assign a trainable offset $\boldsymbol{\Delta}\mathbf{v}_i$ to each vertex $\mathbf{v}_i \in \mathcal{V}$. We use differentiable rendering [22] to optimize these offsets by back-propagating the image-space loss gradients following [32]. In contrast, mesh faces are not differentiable and cannot be optimized via back-propagation in the same way. To address this problem, we propose an iterative mesh refinement algorithm, which is inspired by the Iteratively Reweighted Least Squares (IRLS) algorithm [19]. The key idea is to adaptively adjust face density based on previous training errors. During training, we re-project the 2D pixel-wise rendering errors from Equation 6 to the corresponding mesh faces and accumulate face-wise errors. After a certain number of iterations, we sort all face errors E_{face} and determine two thresholds:

$$e_{\text{subdivide}} = \text{percentile}(E_{\text{face}}, 95), \tag{9}$$

$$e_{\text{decimate}} = \text{percentile}(E_{\text{face}}, 50), \tag{10}$$

Faces with error above $e_{\text{subdivide}}$ are mid-point subdivided [12] to increase face density, while faces with error below e_{decimate} are decimated and remeshed to reduce face density. After the mesh updating, we reinitialize the vertex offsets and face errors and continue the training. This process is repeated several times until stage 1 finishes.

Unbounded scene. Without loss of generality, we are able to model forward-facing [29] and large-scale unbounded scenes [3]. We divide the scene into multiple geometrically growing regions $[-2^k, 2^k]^3, k \in \{0, 1, 2, \cdots\}$ similar to Instant-NGP [31]. Each region exports a separate mesh, with the overlapping part automatically removed to form the whole scene's geometry. Since the outer regions usually lack details compared to the center region ($k = 0$), we also decrease the marching cubes resolution as k increases. The iterative mesh refining only considers the center region since the outer regions are relatively simpler in geometry.

Loss function. To prevent abrupt geometry, we apply a Laplacian smoothing loss $\mathcal{L}_{\text{smooth}}$ [33] on the mesh. In addition, we regularize the vertices offset with an L2 loss:

$$\mathcal{L}_{\text{offset}} = \sum_i (\boldsymbol{\Delta}\mathbf{v}_i)^2, \tag{11}$$

This ensures that the vertices do not move too far from their original positions.

3.3. Mesh Exportation

The goal of our framework is to export a surface mesh with textures that are compatible with commonly used 3D hardware and software. We currently have a surface mesh $\mathcal{M}_{\text{fine}}$ from stage 2, but the appearance is still encoded in a 3D color grid. To extract the appearance as texture images, we first unwrap the UV coordinates of $\mathcal{M}_{\text{fine}}$ using XAtlas [52]. Subsequently, we bake the surface's diffuse color $\mathbf{c}_d$ and specular features $\mathbf{f}_s$ into two separate images, I_d and I_s, respectively.

Real-time rendering. Our exported mesh can be efficiently accelerated and rendered in real-time, as a conventional textured mesh. The diffuse texture I_d can be interpreted as an RGB texture and rendered in most OpenGL-enabled devices and 3D software packages (*e.g.*, Blender [13] and Unity [16]). To render the specular color, we adopt the approach proposed in MobileNeRF [11]. We export the weights of the small MLP$_2$ and incorporate them into a fragment shader. This custom shader enables the addition of the specular term to the diffuse term, allowing for view-dependent effects in real-time.

Mesh manipulation. Similar to a conventional textured mesh, the mesh we export can be readily modified and edited in terms of its geometric and visual attributes. Additionally, it facilitates the combination of multiple exported meshes, as can be observed in Figure 1.

4. Experiment

4.1. Implementation Details

In the first stage, we train for $30,000$ steps, with each step evaluating approximately 2^{18} points. An exponentially decayed learning rate schedule ranging from 0.01 to 0.001 is employed. Specifically, during the initial $1,000$ steps, training solely employs the diffuse color to encourage the appearance factorization. For the second stage, we train additional $10,000$ to $30,000$ steps based on convergence, and set the learning rate for vertex offsets to 0.0001. The Adam [21] optimizer is utilized for both stages. The coarse mesh is extracted at a resolution of 512^3 with a density threshold of 10 by Marching Cubes. We cull the faces invisible from all training camera poses, and decimate the total face number to $30,000$. We maintain a density grid to facilitate ray pruning, following the approach proposed in Instant-NGP [31]. All experiments are conducted on a single NVIDIA V100 GPU. Please refer to the supplementary materials for more details.

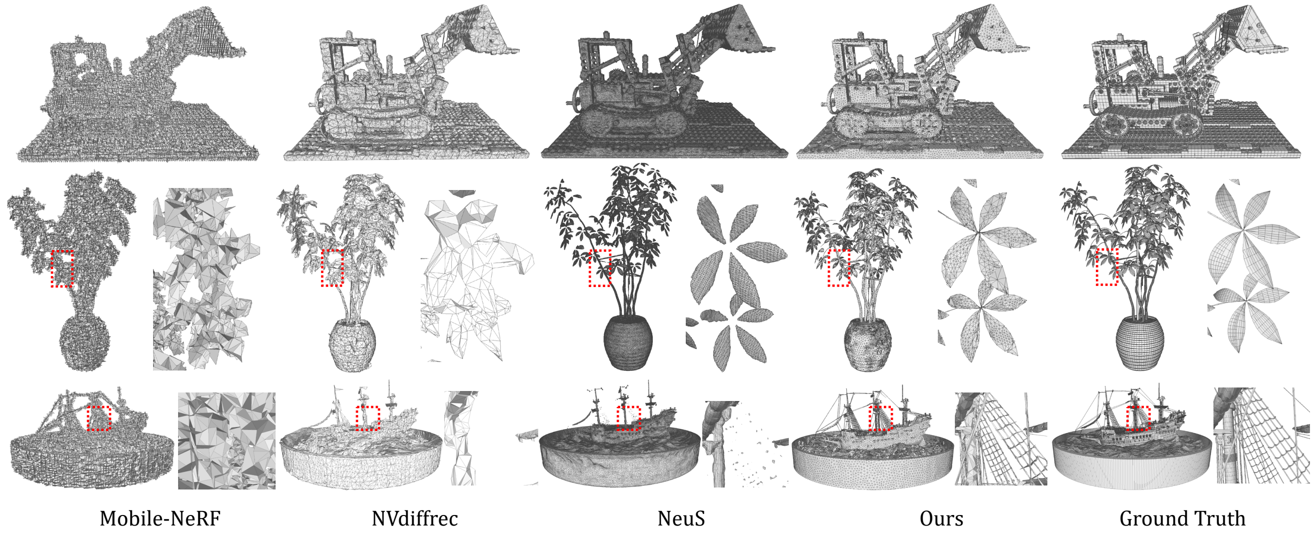

Figure 5: **Surface reconstruction quality on NeRF-synthetic dataset**. Our method achieves superior mesh reconstruction quality compared to previous methods, especially on thin structures with complex topology. We decimate meshes from NeuS [46] to 25% of the original faces since they are too dense to visualize.

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeuS [46]	**3.95**	6.68	2.84	8.36	6.62	**4.10**	**2.99**	9.54	5.64
NVdiffrec [32]	4.13	8.27	5.47	7.31	**5.78**	4.98	3.38	25.89	8.15
Ours (coarse mesh)	5.64	7.95	5.86	7.59	7.12	5.62	6.32	10.49	7.07
Ours (fine mesh)	4.45	**5.93**	**2.50**	**5.63**	5.81	4.46	3.40	**8.29**	**5.06**

Table 1: **Chamfer Distance ↓ (Unit is 10^{-3})** on the NeRF-synthetic dataset compared to the ground truth meshes.

	NeRF-synthetic		LLFF		Mip-NeRF 360	
	#V	#F	#V	#F	#V	#F
Ground Truth	631	873	-	-	-	-
NeuS [46]	1020	2039	-	-	-	-
NVdiffrec [32]	75	80	-	-	-	-
MobileNeRF [11]	494	224	830	339	1436	609
Ours (coarse mesh)	151	300	231	455	446	886
Ours (fine mesh)	200	192	397	446	718	816

Table 2: **Number of Vertices and Faces ↓ (Unit is 10^3).** Our method uses relateively fewer vertices and faces on the NeRF-synthetic dataset with enhanced mesh quality.

	NeRF-synthetic		LLFF		Mip-NeRF 360	
	Disk	Memory	Disk	Memory	Disk	Memory
SNeRG [18]	86.75	2707.25	337.25	4312.13	-	-
MobileNeRF [11]	125.75	538.38	201.50	759.25	344.60	1080.00
Ours	73.53	226.63	124.84	291.50	186.84	411.33

Table 3: **Disk Storage and GPU Memory Usage ↓ (Unit is MB).** We measure the size of exported models and GPU memory usage in rendering.

Datasets. We experiment on three datasets to verify the effectiveness and generalization ability of our method: 1) NeRF-Synthetic [30] dataset contains 8 synthetic scenes. 2) LLFF [29] dataset contains 8 realitic forward-facing scenes. 3) Mip-NeRF 360 [3] dataset contains 3 publicly available realistic unbounded outdoor scenes. Our method generalize well to different types of datasets and reconstruct faithful mesh even for challenging unbounded scenes.

4.2. Comparisons

We mainly compare against methods that exports textured meshes, like MobileNeRF [11] and NVdiffrec [32]. Since our method also involves a NeRF stage, we compare against several volumetric NeRF methods [30, 18] for competence.

4.2.1 Mesh Quality

Surface reconstruction. The lack of ground truth meshes for realistic scenes makes it challenging to measure surface reconstruction quality. As such, we primarily compare results on synthetic datasets, as done in NVdiffrec [32]. We provide qualitative assessments of the extracted meshes produced by different methods, as shown in Figure 5. Specifically, we focus on thin structures such as dense foliage and rope net. Our method successfully reconstructs these structures with high fidelity, while other methods fail to reconstruct the complex geometry accurately. Additionally, our method produces meshes with more order and neatness, similar to human-made ground truths.

To quantify the surface reconstruction quality, we employ the Chamfer Distance (CD) metric. However, as the ground truth meshes may not be surface meshes (*e.g.*, the Lego mesh is actually made up of many small bricks), we cast rays from the test cameras and sample 2.5M points from these ray-surface intersections per scene. In Table 1, we present the averaged CD for all scenes, demonstrating that our method achieves the best results. We note that our approach performs particularly well on scenes with complex topology, such as ficus, ship, and lego. However, our method performs slightly worse on scenes with lots of non-lambertian surfaces, such as materials. This limitation arises from the relatively small capability of our appearance network, as our model often attempts to simulate such lighting effects by tweaking mesh vertices, resulting in incorrect geometry.

Mesh size. We also evaluate the practical applicability by comparing the number of vertices and faces in the exported meshes, as shown in Table 2. Furthermore, we measure the disk storage and GPU memory usage required for rendering the exported meshes, as presented in Table 3. For fair comparison, the mesh file format is uncompressed OBJ & MTL, the texture is PNG, and other metadata is stored in JSON format. Compared to the ground truth and MobileNeRF [11] on the NeRF-synthetic dataset, our exported meshes contain fewer vertices and faces. This is because the iterative mesh refining process can increase the number of vertices to enhance surface details, while simultaneously reducing the number of faces to control mesh size.

4.2.2 Rendering Quality

We present the results of our rendering quality comparison in Table 4. We observe a decrease in rendering quality when distilling from NeRF (volume) to mesh. Specifically, we find that the smoothness regularization term $\mathcal{L}_{\text{smooth}}$ plays a crucial role in maintaining a balance between surface smoothness and rendering quality. Disabling this regularization term leads to better rendering quality at the expense of surface quality (detailed in Section 4.4). We demonstrate that our mesh-based approach yields superior rendering quality compared to NVdiffrec [32], which is currently the state-of-the-art for surface mesh reconstruction. Furthermore, our approach generalizes well to forward-facing and unbounded scenes, while NVdiffrec [32] is only capable of reconstructing single objects. In contrast, MobileNeRF [11] exports grid-like meshes that lack smoothness and may not align well with object surfaces. These meshes rely on texture transparency to carve out the surface. Although our smooth meshes exhibit worse rendering quality, our meshes without the smoothness regularization term achieve comparable performance. Figure 6 presents a visualization of our meshes' rendering quality and compares them with related methods.

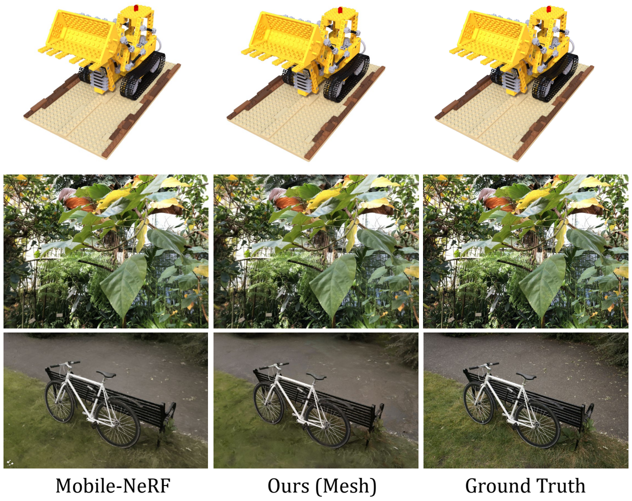

Figure 6: **Visualization of rendering quality**. We achieve comparable rendering quality on different datasets.

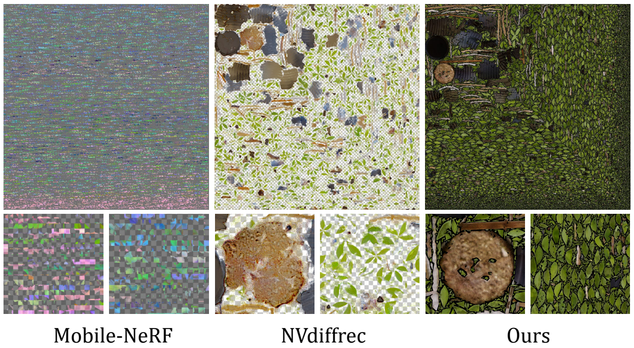

Figure 7: **Visualization of texture images**. We show that our textures are more compact and intuitive due to the enhanced surface quality.

In Figure 7, we also present the texture images exported by different methods. We demonstrate that our high-quality surface meshes result in texture images that are more compact and intuitive than those generated by other methods.

4.3. Efficiency

Our framework demonstrates high efficiency in both training and inference stages. A single NVIDIA V100 GPU

	category	NeRF-synthetic			LLFF			Mip-NeRF 360		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [30]	volume	31.00	0.947	0.081	26.50	0.811	0.250	-	-	-
Ours (volume)		30.79	0.951	0.077	25.92	0.807	0.242	22.18	0.521	0.496
NVdiffrec [32]	surface	29.05	0.939	0.081	-	-	-	-	-	-
Ours (mesh)	mesh	29.67	0.940	0.072	24.40	0.757	0.280	22.27	0.483	0.481
MobileNeRF [11]	non-surface	30.90	0.947	0.062	25.91	0.825	0.183	23.06	0.527	0.434
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)	mesh	30.91	0.947	0.068	25.12	0.778	0.271	22.75	0.529	0.456

Table 4: **Rendering Quality Comparison.** We report PSNR, SSIM, and LPIPS on different datasets, and compare against methods from different categories. We achieve comparable performance for volumetric and non-surface mesh, and better performance for surface mesh.

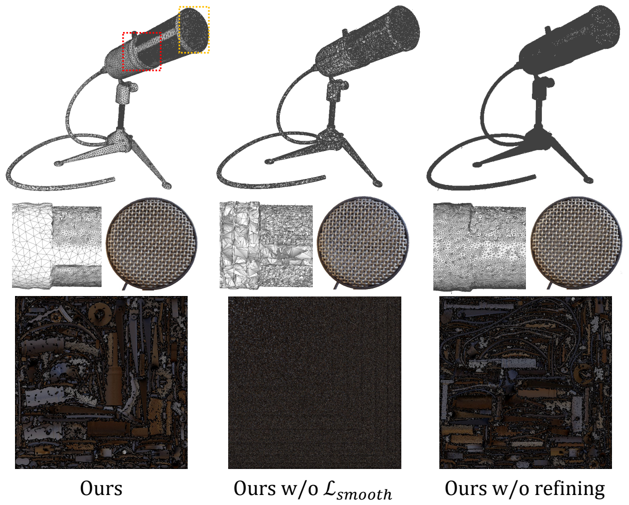

Figure 8: **Qualitative Ablation.** We visualize the mesh structure and texture images under different settings.

with 16GB memory takes roughly 1 hour for the training of the two stages and mesh exportation per scene. In contrast, other competing methods often require several hours [32] or even days [11], with higher hardware demands to complete similar tasks. Furthermore, the exported meshes are lightweight, allowing for real-time rendering on OpenGL-enabled devices, including mobile devices.

4.4. Ablation Studies

In Figure 8 and Table 5, we conduct an ablation study focusing on the geometry optimization stage. Specifically, we compare the full model against variants that exclude either the smoothness regularization or the iterative mesh refining process. The results indicate that: 1) When the smoothness regularization is removed, the resulting surface mesh displays a better rendering quality, but exhibits irregularities and self-intersections. Moreover, the mesh size increases due to the inability of the iterative mesh refining process to work well with such irregular surfaces. These irregular faces also lead to poor UV quality and messy texture images. 2) When the iterative mesh refining is removed, the face density becomes almost uniform, resulting in a larger mesh size and slightly inferior rendering quality. This is because face density cannot be adaptively adjusted based on the re-projected rendering errors.

	#V	#F	Size (MB)	PSNR
Ours	58,649	116,698	54.8	31.30
Ours w/o $\mathcal{L}_{\text{smooth}}$	202,656	396,385	133.0	32.57
Ours w/o refining	150,276	300,000	74.8	31.06

Table 5: **Quantitative Ablation.** We report the mesh statistics and PSNR on the Mic scene.

5. Limitations and Conclusion

Although our method has shown promising results, it still has several limitations. Due to the difficulty of estimating unknown lighting conditions from images without compromising reconstruction quality [57], we have chosen to bake illumination into textures, which consequently restricts our ability to perform relighting. Our relatively small appearance network also makes it challenging to learn complex view-dependent effects, which can result in lower surface quality in such regions. These choices were made intentionally to maintain pipeline efficiency. In the future, we hope to address these limitations by leveraging better appearance modeling techniques. Lastly, similar to other mesh-based methods [32, 11], we perform a single-pass rasterization and are unable to handle semi-transparency.

In summary, we present an efficient framework that can reconstruct textured surface meshes from multi-view RGB images. Our approach utilizes NeRF for coarse geometry and appearance initialization, subsequently extracts and enhances a polygonal mesh, and ultimately bakes the appearance into texture images for real-time rendering. The reconstructed meshes demonstrate an enhanced surface quality, particularly for thin structures, and are convenient to manipulation and editing fow downstream applications.

References

[1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. *arXiv preprint arXiv:2301.02238*, 2023. 2

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5855–5864, 2021. 1, 2

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2, 5, 6

[4] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020. 2

[5] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *ICCV*, 2021. 2

[6] Mark Boss, Andreas Engelhardt, Abhishek Kar, Yuanzhen Li, Deqing Sun, Jonathan T Barron, Hendrik Lensch, and Varun Jampani. Samurai: Shape and material from unconstrained real-world arbitrary image collections. *arXiv preprint arXiv:2205.15768*, 2022. 1, 3

[7] Junli Cao, Huan Wang, Pavlo Chemerys, Vladislav Shakhrai, Ju Hu, Yun Fu, Denys Makoviichuk, Sergey Tulyakov, and Jian Ren. Real-time neural light field on mobile devices. *arXiv preprint arXiv:2212.08057*, 2022. 2

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 1, 2, 3, 4

[9] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *NeurIPS*, 32, 2019. 2

[10] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, and Sanja Fidler. Dib-r++: learning to predict lighting and material with a hybrid differentiable renderer. *NeurIPS*, 34:22834–22848, 2021. 2

[11] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022. 1, 3, 5, 6, 7, 8

[12] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 5

[13] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 5

[14] François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces geometry with patch warping. In *CVPR*, pages 6260–6269, 2022. 3

[15] Qiancheng Fu, Qingshan Xu, Yew-Soon Ong, and Wenbing Tao. Geo-neus: geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *arXiv preprint arXiv:2205.15848*, 2022. 3

[16] John K Haas. A history of the unity game engine. 2014. 5

[17] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. *arXiv:2206.03380*, 2022. 2

[18] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *ICCV*, 2021. 6

[19] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977. 5

[20] Krishna Murthy Jatavallabhula, Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. Kaolin: A pytorch library for accelerating 3d deep learning research. *arXiv preprint arXiv:1911.05063*, 2019. 2

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[22] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 39(6), 2020. 2, 5

[23] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*, 2022. 2

[24] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *CVPR*, pages 2916–2925, 2018. 2

[25] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, pages 7708–7717, 2019. 2

[26] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *ICCV*, pages 5773–5783, 2021. 2

[27] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. *arXiv preprint arXiv:2211.14173*, 2022. 3

[28] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, 21(4):163–169, 1987. 2, 3, 5

[29] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 2019. 5, 6

[30] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3, 6, 8

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 1, 2, 3, 5

[32] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *CVPR*, 2022. 1, 2, 3, 5, 6, 7, 8

[33] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, 2006. 5

[34] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, pages 5865–5874, 2021. 2

[35] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 3

[36] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *ICCV*, 2021. 2

[37] Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *arXiv preprint arXiv:2302.12249*, 2023. 2

[38] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2, 3

[39] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *NeurIPS*, 34:6087–6101, 2021. 2

[40] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, pages 7495–7504, 2021. 2

[41] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *CVPR*, 2022. 2, 3, 4

[42] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. *arXiv preprint arXiv:2205.14870*, 2022. 2

[43] Itsuki Ueda, Yoshihiro Fukuhara, Hirokatsu Kataoka, Hiroaki Aizawa, Hidehiko Shishido, and Itaru Kitahara. Neural density-distance fields. In *ECCV*, pages 53–68. Springer, 2022. 3

[44] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 2

[45] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. *arXiv preprint arXiv:2112.05139*, 2021. 2

[46] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 2, 3, 6

[47] Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. Voxurf: Voxel-based efficient and accurate neural surface reconstruction. *arXiv preprint arXiv:2208.12697*, 2022. 3

[48] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *CVPR*, pages 7119–7128, 2021. 2

[49] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, October 2021. 2

[50] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *NeurIPS*, 34:4805–4815, 2021. 2, 3

[51] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P.Srinivasan, Richard Szeliski, Jonathan T.Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv preprint arXiv:2302.14859*, 2023. 1, 3

[52] Jonathan Young. Xatlas, 2021. 5

[53] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[54] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *arXiv preprint arXiv:2206.00665*, 2022. 2, 3

[55] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[56] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[57] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM TOG*, 40(6):1–18, 2021. 2, 3, 8