LTM: Lightweight Textured Mesh Extraction and Refinement of Large Unbounded Scenes for Efficient Storage and Real-time Rendering

Jaehoon Choi [1] Rajvi Shah [2] Qinbo Li [2] Yipeng Wang [2] Ayush Saraf [2] Changil Kim [2]
Jia-Bin Huang [1,2] Dinesh Manocha [1] Suhib Alsisan [2] Johannes Kopf [2]
[1]University of Maryland [2]Meta

Abstract

Advancements in neural signed distance fields (SDFs) have enabled modeling 3D surface geometry from a set of 2D images of real-world scenes. Baking neural SDFs can extract explicit mesh with appearance baked into texture maps as neural features. The baked meshes still have a large memory footprint and require a powerful GPU for real-time rendering. Neural optimization of such large meshes with differentiable rendering pose significant challenges. We propose a method to produce optimized meshes for large unbounded scenes with low triangle budget and high fidelity of geometry and appearance. We achieve this by combining advancements in baking neural SDFs with classical mesh simplification techniques and proposing a joint appearance-geometry refinement step. The visual quality is comparable to or better than state-of-the-art neural meshing and baking methods with high geometric accuracy despite significant reduction in triangle count, making the produced meshes efficient for storage, transmission, and rendering on mobile hardware. We validate the effectiveness of the proposed method on large unbounded scenes from mip-NeRF 360, Tanks & Temples, and Deep Blending datasets, achieving at-par rendering quality with 73× reduced triangles and 11× reduction in memory footprint.

1. Introduction

Photorealistic reconstruction and rendering of real-world objects and scenes is a longstanding problem of importance with several applications in computer vision, robotics and AR/VR. In recent years, Neural Radiance Fields (NeRFs) [1, 2, 38, 39] have been quite successful at novel view synthesis. However, the scene representation is learned as an implicit volumetric representation that is expensive to evaluate. This is a crucial limitation for many applications, such as Augmented and Virtual Reality (AR/VR), often requiring real-time rendering with low memory and compute power.

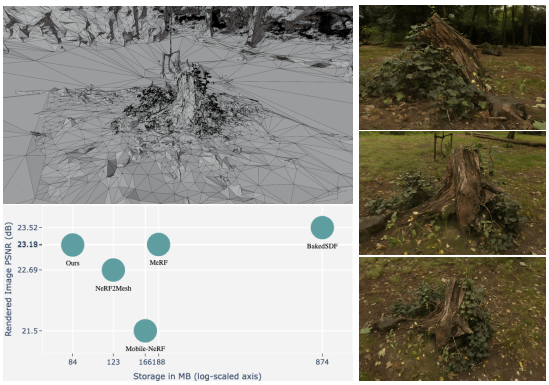

Figure 1. Example results using LTM showing mesh geometry (top-left) and novel-view synthesis (right). Bottom right plot compares performance of different methods on mip-NeRF360 outdoor dataset. Our method achieves comparable rendering quality to BakedSDF while reducing the mesh size considerably.

Since modern graphics engines are highly optimized to rasterize triangle meshes, researchers have started exploring ways to extract explicit mesh representations from volumetric functions.

MobileNeRF [5] uses a polygon mesh with texture maps storing feature vectors and opacity. Although their rendering quality is good, the mesh quality is far from an ideal surface. While NeRF2Mesh [50] and NeRFMeshing [44], based on volumetric density fields, excel at generating highly detailed geometry, they often result in noisy and bumpy surfaces and numerous floaters, especially in planar regions, which are common in large-scale scenes. In contrast, neural Signed Distance Function (SDF) methods [42, 52, 54] typically excel in reconstructing accurate surface geometry. However, they may not be able to capture small or detailed geometry, and their rendering quality is inferior to volumetric density-field methods. BakedSDF [55] overcomes these limitations by applying mip-NeRF 360 techniques to train SDF representation, though the resulting mesh representation has a high memory overhead.

Previous neural SDF methods that rely on the Marching Cube algorithm [34] result in meshes with high storage overhead. This results in a significant challenge for practi-

cal applications on mobile devices, where low disk storage and fast data transfer are essential. Instead of relying on the Marching Cubes, some methods [11, 28, 48, 49] propose differentiable mesh reconstruction methods. These methods aim to extract an isosurface from an implicit function and can be seamlessly integrated with differentiable rendering approaches [16, 17]. However, most methods primarily concentrate on a single object with ideal implicit surface functions, unsuited for large unbounded real-world scenes [2, 55]. Reconstructing an ideal mesh for the unbounded large-scale scene is still an open problem, often resulting in non-manifold geometry or missing surfaces. Consequently, the renderings can result in visual artifacts due to the incompleteness of the mesh geometry.

Main Results: In this paper, our goal is to achieve a realistic rendering with a compact mesh representation that optimally utilizes disk storage for unbounded large-scale scenes, all while preserving geometric quality. Our method initially trains neural fields for geometry and appearance separately. For geometry reconstruction, we adopt the BakedSDF approach [55] to train an SDF in a contracted space [2]. Regarding the appearance field, we utilize hash-based encoding [39] with two shallow MLPs to represent diffuse and specular colors. The SDF and appearance fields are compact representations for reconstructing a high-quality mesh and achieving realistic rendering.

Given the initial geometry representations, we perform mesh extraction and decimation. As unbounded large-scale scenes occupy a large amount of disk storage, our method focuses on generating a lightweight mesh without compromising geometric details. We adaptively extract the mesh for the central and background regions in the contracted space, respectively. Subsequently, we modify a mesh decimation [14] designed for large-scale scenes. Our method eliminates redundant triangles corresponding to simple geometric features, such as planar surfaces, while preserving complex geometric features.

Since the appearance field is trained by volumetric rendering, the initial model is unsuitable for rasterization-based rendering. Moreover, the mesh from the previous step still fails to accurately represent areas with highly-detailed geometry (thin structures) or semi-transparent objects. These mesh defects hurt rendering quality. Thus, we use differentiable rendering [27] and utilize image-space supervision to refine the mesh surface by rectifying these mesh defects while preserving geometric accuracy. Due to the highly simplified mesh and various regularization techniques, our approach can deform the mesh topology without causing severe mesh distortion, a common issue in gradient-based mesh optimization [41, 43]. Compared to mesh-based neural rendering methods [5, 44, 50, 55], our approach results in lower storage overhead while providing comparable rendering quality. To sum up, our contributions include:

- We effectively utilize classical mesh decimation techniques to extract compact geometry from neural SDFs of large unbounded scenes.
- We introduce a new approach for joint optimization of appearance and vertex deformation that preserves geometric details and improves rendering quality.
- We achieve a trifecta of precise geometry, efficient storage, and rendering quality comparable to the state of the art on diverse captured environments such as mip-NeRF 360, Tanks & Temples, and the Deep Blending dataset.

2. Related Work

Neural Rendering Neural Fields achieve remarkable photo-realistic rendering quality. One of the most successful approaches is Neural Radiance Fields (NeRFs) [38], which adopts volumetric implicit representations and trains a radiance field through ray marching. Mip-NeRF [1, 2] presents casting a conical frustum into the scene space and extends this integrated positional encoding to unbounded scenes by using a novel contraction function. Instead of learning volumetric density, neural surface reconstruction methods [6, 42, 52, 54] propose learning a Signed Distance Function (SDF) for surface reconstruction via volumetric rendering. Furthermore, several methods have been proposed to accelerate NeRFs by utilizing efficient data structure [3, 10, 39], although achieving real-time rendering for high-resolutions remains a challenge. Alternatively, to enhance rendering speed, "baking" based methods [5, 19, 44, 45, 50, 55] typically precompute and store a trained NeRF into a more efficient representation. SNeRG [19] adopts a deferred shading model [8] and bakes trained NeRFs into sparse voxel grid representations with neural features. In particular, the mesh is widely used as geometry representation and bakes neural fields into its texture. MobileNeRF [5] builds on a mesh rasterization pipeline for efficient rendering on mobile architecture. NeRF2Mesh [50] introduces a two-step framework for NeRF training and textured mesh extraction. NeRFMeshing [44] proposes a novel method to produce a truncated SDF from NeRFs and extracts the accurate mesh. BakedSDF [55] enables us to reconstruct high-quality mesh and bake it into a small appearance model based on spherical Gaussians for fast rendering. DNMP [35] and NeuRaS [32] adopt mesh-based geometry and rasterization-based rendering for large driving scenes.

Mesh Processing Traditional graphics research has developed effective techniques for simplifying meshes, based on vertex decimation [46, 47], edge collapse [9, 12–14, 20] and appearance preservation [7]. Vertex decimation methods prioritize vertices based on shape heuristics and remove the least important ones. Edge collapse methods rank the edges based on the cost and subsequently contract pairs of vertices. QSlim [12, 13] introduces the concept of the

Quadric Error Metric (QEM), which is defined as the sum of quadrics representing the distance of a point from a set of planes. Following the edge collapse, they utilize this QEM to identify which edges should be removed while preserving geometric topology. Lindstrom et al. [29, 30] use vertex clustering [46] on a uniform grid and accumulate a quadric error matrix to the occupied grid cell. Wang et al. [51] employ RGB-D sequences to reconstruct a lightweight mesh.

Differentiable Rendering and Mesh Optimization Recently, many differentiable rendering techniques [23] have emerged, aimed at facilitating end-to-end optimization of the rendering process by employing useful gradients. In particular, several works [4, 33] focus on approximating the rasterization stage of the rendering to compute meaningful gradients. Furthermore, modular differentiable renderers [21, 22, 27] exist, allowing users to customize and modify specific functions. Differentiable mesh reconstruction techniques [11, 28, 48] can build a mesh through gradient-based optimization. In particular, differentiable rendering [16, 17, 49] can offer 2D supervision to refine and enhance the geometric quality of the mesh. Some works [16, 17] adopt an inverse rendering approach, leveraging physics-based rendering to synthesize shapes. Nevertheless, gradient-based mesh optimization [41] is highly sensitive to initialization and prone to significant mesh distortion. Some methods [31, 41, 43] incorporate a novel ADAM optimizer and employ strong regularization techniques. However, this research is currently limited to generating single objects and is challenging to apply to large-scale scenes.

3. Our Method

Given multi-view images and corresponding camera poses, we aim to reconstruct the texture and geometry of unbounded large-scale scenes. Fig. 2 shows the three main stages of our pipeline: Initialization, Mesh Processing, and Optimization. In the first stage, we employ volumetric rendering to train neural implicit representation for geometry f_{sdf} and appearance f_{color}. Subsequently, we leverage the geometry representation f_{sdf} to extract a high-resolution mesh via Marching Cubes [34] and reduce the number of triangles without causing geometry distortion. Next, we utilize differentiable rendering [27] to jointly optimize the explicit 3D mesh M and implicit appearance field f_{color} obtained from the first stage and refine the mesh surface.

3.1. Initialization of Geometry and Appearance

In this stage, we first learn neural implicit representations for both appearance and geometry. A neural radiance field [38] is an implicit representation of a 3D scene volume, a mapping from a 3D position x and a ray viewing direction $d \in \mathbb{R}^3$ to an RGB color c and volume density ρ. In practice, the networks modeled by MLPs are used to generate this

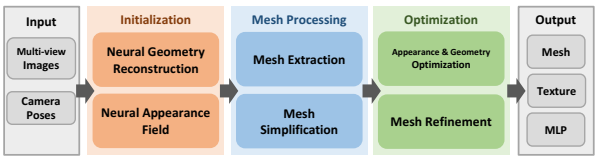

Figure 2. **Overview of Our Pipeline**. It consists of three steps: Initialization (Sec. 3.1), Mesh Processing (Sec. 3.2), and Optimization (Sec. 3.3). Our method focuses on baking the initial appearance and geometry representation, trained by volumetric rendering, into an explicit mesh and a memory-efficient texture conducive to real-time rendering. We employ classical mesh decimation to minimize storage requirements, introduce the optimization stage to facilitate rasterization-based rendering and refine the detailed mesh structure.

volume density and view-dependent color:

$$\rho = f_{density}(x), \quad c = f_{color}(x, d). \tag{1}$$

To render the color of a pixel, we first compute the 3D points t_i along the viewing ray associated with each pixel p as $x(t) = o + td$. Then, estimated volume densities ρ_i and colors c_i are utilized to approximate a volume rendering equation [37] using numerical quadrature:

$$C = \sum_i T_i \alpha_i c_i, \quad T_i = \exp(-\sum_{j<i} \rho_i \delta_i), \tag{2}$$

where T_i is the accumulated transmittance and $\alpha_i = 1 - \exp(-\rho_i \delta_i)$ is the discrete opacity of the point. $\delta_i = t_i - t_{i-1}$ is the distance between two neighboring points along the ray. To deal with large unbounded scenes, mip-NeRF 360 [2] introduces a proposal MLP for efficient sampling and space contraction to represent an unbounded scene. The scene contraction function f (in Eq. 3) maps the unbounded points into bounded space:

$$f(x) = \begin{cases} x & ||x|| \leq 1, \\ (2 - \frac{1}{||x||})\frac{x}{||x||} & ||x|| \geq 1. \end{cases} \tag{3}$$

Appearance Similar to previous works [5, 19, 50], we decompose an appearance field f_{color} into shallow MLPs to learn diffuse colors c_d and specular colors c_s separately. Particularly, the view-dependent effects are modeled by a small MLP that can be easily integrated into a fragment shader. In order to ensure the MLPs are able to model an unbounded scene well, we adopt strategies similar to the ones introduced in mip-NeRF360 work, specifically, scene contraction, proposal MLP, and regularization for the background. We train both MLP networks f_{color} and $f_{density}$ with a multi-resolution hash positional encoding [39] using the volume rendering objective in Eq. 2. The appearance field trained in this stage is further optimized to facilitate rasterization-based rendering, replacing the computationally expensive ray marching. This optimization is described later in Section 3.3.

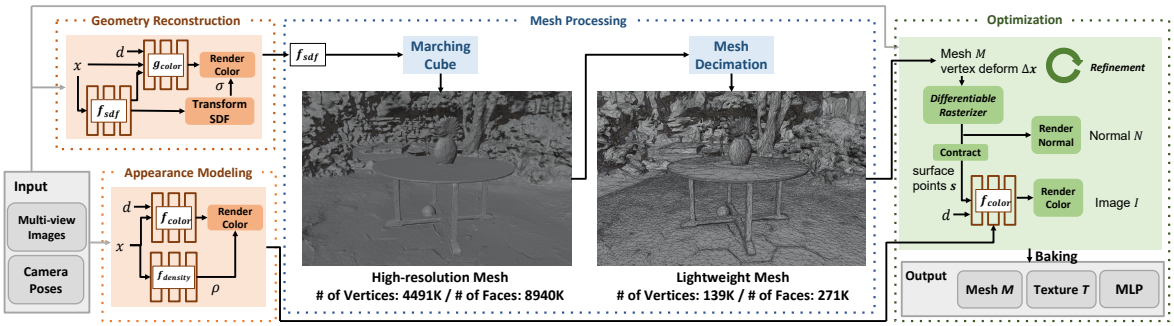

Figure 3. **Our Approach:** In Section 3.1, we start training both the appearance f_{color} and geometry f_{sdf} representations for a large-scale scene using volumetric rendering (shown in orange). Then, in Section 3.2, our method extracts the mesh from f_{sdf} and significantly simplifies its structure, resulting in the lightweight mesh M (shown in blue). In Section 3.3, we jointly train vertex deformation Δx and f_{color} computed from appearance modeling.

Geometry Initial geometry modeling in our method is similar to BakedSDF [55]. The core mechanism for neural surface reconstruction was introduced in VolSDF [54] for small objects. BakedSDF shows that by leveraging the efficient sampling and contraction ideas introduced in mip-NeRF360 [2], VolSDF [54] can be extended to large unbounded scenes. We follow this and jointly train two MLPs using the volumetric rendering objective, (i) f_{sdf} - representing geometry of a scene as a signed distance function, and (ii) g_{color} - a color prediction network, as shown in the *Geometry Reconstruction* block in Fig.3.

Following VolSDF, we define the volume density $\sigma(x)$ as a function of surface geometry f_{sdf}, namely, $\sigma(x) = \alpha\Psi_\beta(f_{sdf}(x))$, where Ψ_β is the Cumulative Distribution Function (CDF) of the Laplace distribution with zero mean and β is the learnable scaling parameter. The Eikonal loss [15] to regularize the f_{sdf} is defined as,

$$L_{eik} = (\parallel \nabla f_{sdf}(x) \parallel_2 -1)^2. \tag{4}$$

The color MLP g_{color}, in addition to the position and viewing direction, also takes the predicted surface normal and a 256-dimensional geometric feature as inputs. It is important to note that the sole purpose of color MLP g_{color} is geometry initialization and we discard it after this step. This is not to be confused with the shallow decomposed color MLP f_{color} trained specifically to model appearance that is also jointly refined with mesh vertices in the later stage.

3.2. Mesh Extraction and Decimation

In this stage, we first extract a mesh by using Marching Cubes (MC) [34] algorithm on learned SDF (f_{sdf} evaluated over a uniform grid) and simplify the extracted mesh using a geometry-aware quadric error based mesh decimation. While this approach works pretty well for extracting meshes of well-captured small-scale objects and scenes, it presents unique challenges for large-scale scenes. Meshing neural SDFs of large-scale scenes with marching cubes can often result in spurious surfaces arising from (i) insufficient grid resolution to model complex geometry of nearby objects, and (ii) poorly learned geometry of distant objects due to insufficient view-points observing such distant regions. Furthermore, since our SDF f_{sdf} is trained in the contracted space, the resulting mesh is also defined in the contracted space. When we *uncontract*, i.e. warp the mesh vertices back into the original Euclidean space, it further exacerbates the geometry distortion in the background region.

We aim to preserve the detailed geometry in central (foreground) regions while achieving a coarser but less noisy reconstruction of distant (background) regions, within a small triangle budget. We achieve this by treating foreground and background regions distinctly during both extraction and simplification steps. During MC extraction, we set a finer grid resolution of 2048 and a spatial range of $[-1, 1]^3$ for the central region, and a coarser resolution of 1024 and a spatial range of $[-2, 2]^3$ for the background region. For decimation, we first partition the central and background regions and employ an aggressive mesh decimation in background regions. This helps us transform bumpy and noisy background areas into smooth surfaces while also reducing redundant triangles.

We first apply vertex clustering and then run quadric error based simplification algorithm - QSlim [12, 13] for all clusters. Then, we run edge contraction for each cluster to reach a threshold and repeat this process until convergence. The QSlim algorithm calculates a quadric metric for each vertex to decide whether the two vertices should be merged and aggregates these quadrics during each edge collapse operation. The basic quadric error in QSlim [13] is defined as the point-to-plane distance from vertex $v \in \mathbb{R}^3$ to all planes $p \in \mathbb{R}^4$ of its adjacent faces $N(v)$:

$$E(v) = \sum_{p \in N(v)} (p^T v)^2 = v^T \Big(\sum_{p \in N(v)} pp^T \Big) v = v^T Q_v v. \tag{5}$$

The quadric Q_v is the symmetric 4x4 matrix, and each edge computes this quadric by adding the quadrics of the corresponding two vertices. The most commonly used quadric error is defined solely based on 3D vertex positions v.

Intuitively, the most effective strategy is to represent planar surfaces with a smaller number of large triangles while keeping a consistent number of triangles for complex shapes. Variants that incorporate additional surface attributes, such as normals or textures in calculating the quadric error are proposed in [12]. We experimentally observed that compared to the basic QSlim algorithm, using both the vertex positions $v \in \mathbb{R}^3$ and the corresponding vertex normal $n \in \mathbb{R}^3$ for quadric error is very effective at prioritizing triangle decimation on flat surfaces while retaining them on complex surfaces. The combination of vertex clustering and geometry-aware QSlim helps us in effectively reducing the number of triangles while preserving the original geometry.

3.3. Appearance and Geometry Optimization

In this stage, we optimize the appearance model jointly with mesh vertex deformation and refine the mesh topology using a geometric complexity based criterion.

Joint optimization of appearance and vertex deformation Given the simplified mesh M and the initial appearance field f_{color}, we employ the fast differentiable rasterizer introduced in Nvdiffrast [27] to jointly optimize the mesh geometry and the appearance field. The initial appearance field f_{color} trained using the volume rendering objective (Eq. 1) is agnostic of the surface geometry of the extracted and simplified mesh. To finetune the model to fit the mesh geometry, we rasterize the mesh into each of the training camera poses and obtain surface points s and face normals N interpolated per pixel. Then we apply the contraction function in Eq. 3 to warp the surface points. This is because the appearance field defined in contracted space proves to be more effective in enhancing the rendering quality [2]. The appearance field f_{color} takes these surface points s and viewing direction d as input, rendering an image $\hat{I}_i$. Initially, rendering performance is degraded because the appearance field only utilizes the intersection point between the ray and the surface rather than accumulating all sampled points along the ray. We finetune this appearance field to be compatible with mesh rendering, akin to the functionality of a fragment shader.

Given the positions x of the mesh vertices, we train per-vertex deformation Δx using differentiable rendering [27]. We explore two approaches to optimize vertex deformation: a standard ADAM optimizer [24] and a rotation-invariant ADAM optimizer [43]. The ADAM optimizer is widely employed [16, 50] for deforming mesh topology to enhance rendering performance. While the ADAM optimizer often results in mesh distortions such as flipped triangles and self-intersections, our method mitigates mesh distortion, especially on smooth surfaces, due to the rigorous mesh decimation performed in Section 3.2.

Objective Function To jointly train appearance field and vertex deformation, we minimize the image-space L1 loss $L_{appearance}$ between the rendered image $\hat{I}_i$ of the deformed surface and the groundtruth image I_i for a camera view P_i:

$$L_{appearance} = \sum_i |\hat{I}_i - I_i|. \tag{6}$$

Naively training vertex deformation is highly likely to result in undesirable mesh topology characterized by a disorganized arrangement of triangles, often referred to as a "polygon soup." To prevent this mesh distortion, we employ various regularization methods. We use the Laplacian mesh smoothness loss and the normal consistency loss inspired by [36].

$$L_{lap} = \sum \parallel LV \parallel_2 \tag{7}$$

where $V \in \mathbb{R}^{N\text{x}3}$ (N is the number of vertices) is the matrix of deformed vertex coordinates, indicating $x + \Delta x$, at each training step, and $L \in \mathbb{R}^{N\text{x}N}$ is the graph Laplacian of the mesh [40]. Minimizing the magnitude of a vertex's differential coordinates (LV) reduces its distance from its neighbors' average position. The normal consistency loss is

$$L_{nc} = \sum_{i,j \in F} (1 - n_i \cdot n_j)^2, \tag{8}$$

where n_i and n_j denote the surface normal of two adjacent faces and F is the set of faces sharing a common edge. Additionally, in Section 3.2, a high-resolution mesh is rasterized in the first pass before decimation. The output of this rasterization is used to interpolate face normals per pixel, resulting in a normal map $\hat{N}_i$ for every viewpoint. We calculate the L1 loss between a rendered normal N_i from current mesh M and the $\hat{N}_i$:

$$L_{norm} = \sum_i |\hat{N}_i - N_i|. \tag{9}$$

L_{norm} provides geometry supervision, particularly in planar regions, helping prevent severe mesh distortion when moving vertices. The total loss function L is defined as:

$$L = L_{color} + \lambda_{lap} L_{lap} + \lambda_{nc} L_{nc} + \lambda_{norm} L_{norm}, \tag{10}$$

where λ_* is the weight to balance between the loss terms.

Mesh Refinement Inspired by NeRF2Mesh [50], we adopt a classical mesh refinement approach based on mesh subdivision and decimation during training. The mesh refinement strategy is applied differently to the central and background regions. Instead of rendering error proposed by [50], which can arise from many factors including specularity, we employ a criterion: the vertex deformation values for subdividing triangle faces. The large vertex deformation implies a more substantial triangle budget to accurately capture the intricacies of highly detailed geometry associated

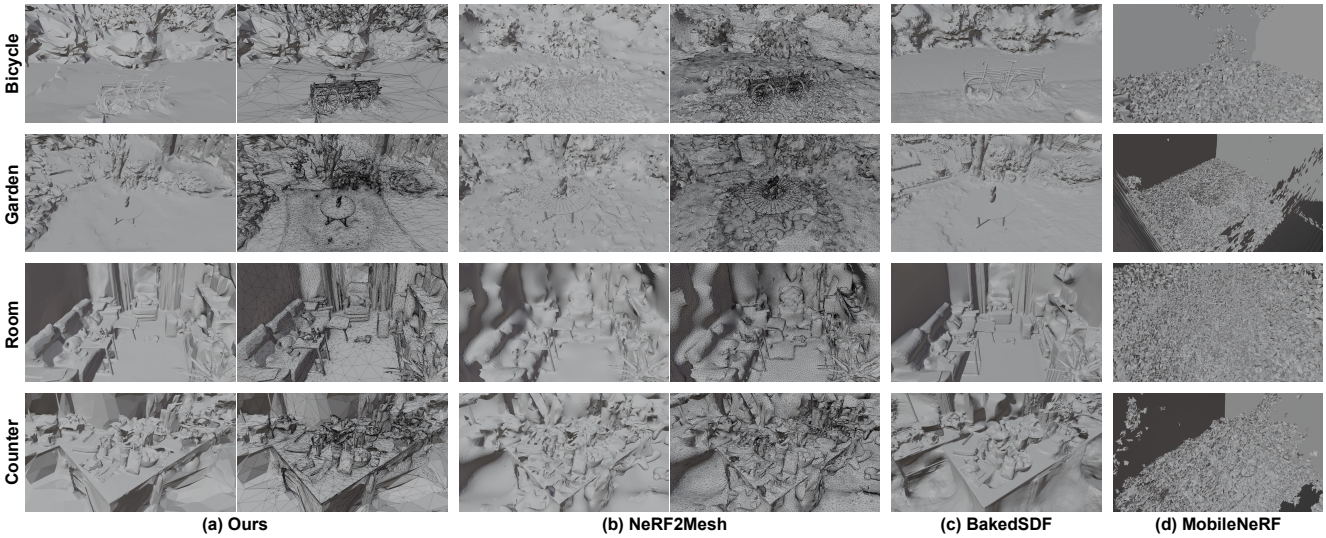

Figure 4. **Visual comparison of Our Method, NeRF2Mesh [50], BakedSDF [55], and MobileNeRF [5].** Our method and NeRF2Mesh show shading mesh and wireframe mesh extracted without texture, respectively.

with those vertices. We sort the absolute values of vertex deformation and determine a 10 % of faces to increase triangles. Then, we apply $\sqrt{3}$ subdivision [26], which adds a vertex at the barycenter of a triangle and connects it to all vertices of this triangle. This scheme enables us to augment the triangle budget for complex geometry and mitigate self-intersection during vertex deformation. Furthermore, the mesh decimation presented in Section 3.2 is applied after every mesh subdivision to remove redundant triangles on simple and flat geometry. For background regions, we examine all training views and check faces intersecting with ray casting. We utilize dilation for all invisible masks and employ mesh decimation to remove redundant faces.

Rotation-Invariant ADAM The rotation-invariant ADAM formulation [43] modifies the update rule in ADAM, resulting in the update step: $\Delta x = \zeta \cdot v \cdot l_{ref}$. Here, ζ is the learning rate, and v is the standard ADAM update step, which is the loss gradient with respect to x. l_{ref} is the coefficient initially set to the edge length incident to x, and it gets updated based on the average v.

4. Experiments

4.1. Experiment Settings

Datasets: To validate the effectiveness of our method, we mainly conduct experiments on seven unbounded, large-scale 360° indoor and outdoor scenes from the mip-NeRF 360 [2] dataset. Also, we validate the generalization ability of our approach on two different datasets, including large-scale scenes: two scenes from the Tanks & Temples [25] dataset and two scenes from the Deep Blending [18] dataset.

Implementation Details: For all datasets, we subsampled every eight frames for test sets following the mip-NeRF 360 paper [2]. For geometry reconstruction in Section 3.1, we use SDFstudio [56] to implement BakedSDF. In Section 3.2, we reduce 95% of the triangles in the central region and 99% of the triangles in the background regions through decimation. In Section 3.3, we run all experiments for 40k iterations, and a learning rate is annealed from 1e-2 to 1e-3. We set the loss weights to λ_{lap}=1e-3, λ_{nc}=1e-3, and λ_{norm}=1e-4. We apply mesh refinement at 8k, 16k, and 24k training steps. Subsequently, we compute diffuse color and specular features using f_{color} to export textures. All these color maps and features are 3-channel .png files. Additional details are provided in the supplementary materials.

4.2. Experimental Results

We compare our proposed method with three different methods: 1) MobileNeRF [5] is the state-of-the-art method, enabling NeRF to achieve real-time rendering of explicit 3D mesh and neural features. 2) NeRF2Mesh [50] employs a volumetric rendering to reconstruct the coarse mesh and refine the mesh by jointly optimizing appearance and geometry. 3) BakedSDF [55] is a recent method for optimizing a neural surface-volume representation for real-time rendering of large unbounded scenes. We also add NeRFMeshing [44], which distills the rendering NeRF into a neural field that represents TSDF (Truncated SDF) and reconstructs the 3D mesh. As BakedSDF and NeRFMeshing have not released public code, we adopt the metrics provided in their original papers. BakedSDF has made its GLB file available. We utilize this file for visualization.

Figure 4 qualitatively compares our method with NeRF2Mesh [50], BakedSDF [55], and MobileNeRF [5]. We visualize the shading meshes extracted without texture, which is the most effective way to visualize geometric differences. Furthermore, we include a wireframe mesh, distinguishing between our method and NeRF2Mesh. Visu-

Method	Mesh	Outdoor (PSNR ↑ / SSIM ↑ / LPIPS ↓)				Indoor (PSNR ↑ / SSIM ↑ / LPIPS ↓)				
		Bicycle	Garden	Stump	Mean	Room	Counter	Kitchen	Bonsai	Mean
Instant-NGP [39]	X	22.1 / 0.49 / 0.49	24.5 / 0.65 / 0.31	23.6 / 0.57 / 0.45	23.4 / 0.57 / 0.42	29.2 / 0.85 / 0.30	26.43 / 0.80 / 0.34	28.5 / 0.82 / 0.25	30.3 / 0.89 / 0.23	28.6 / 0.84 / 0.28
mip-NeRF 360 [2]	X	24.4 / 0.68 / 0.30	26.9 / 0.81 / 0.17	26.4 / 0.74 / 0.26	25.9 / 0.75 / 0.24	31.6 / 0.91 / 0.21	29.5 / 0.89 / 0.20	32.3 / 0.92 / 0.13	33.5 / 0.94 / 0.18	31.7 / 0.92 / 0.18
MobileNeRF [5]	O	21.7 / 0.43 / 0.51	18.8 / 0.59 / 0.36	**23.9** / 0.56 / 0.43	21.5 / 0.53 / 0.43	28.9 / 0.85 / 0.28	25.1 / 0.72 / 0.29	**26.8** / 0.79 / 0.79	23.8 / 0.71 / 0.72	26.2 / 0.77 / 0.52
NeRF2Mesh [50]	O	22.1 / 0.48 / 0.51	23.4 / 0.55 / 0.40	22.5 / 0.54 / 0.46	22.7 / 0.52 / 0.46	25.7 / 0.79 / 0.35	23.9 / 0.71 / 0.35	24.0 / 0.61 / 0.36	25.0 / 0.77 / 0.29	24.7 / 0.72 / 0.34
NeRFMeshing [44]	O	21.1 / - / -	22.9 / - / -	22.6 / - / -	22.2 / - / -	26.1 / - / -	20.0 / - / -	23.6 / - / -	25.6 / - / -	23.8 / - / -
BakedSDF [55]	O	22.0 / **0.57** / **0.37**	**24.9** / **0.75** / **0.21**	23.6 / **0.59** / **0.37**	**23.5** / **0.64** / **0.32**	28.7 / 0.87 / 0.25	**25.7** / **0.81** / 0.28	26.7 / **0.82** / 0.24	27.2 / **0.85** / 0.26	27.0 / **0.84** / 0.26
Ours	O	**22.4** / 0.52 / 0.44	23.5 / 0.64 / 0.30	23.7 / 0.57 / 0.42	23.2 / 0.58 / 0.39	**29.3** / **0.88** / **0.23**	25.1 / 0.77 / **0.27**	26.5 / 0.80 / **0.21**	**27.3** / 0.84 / **0.22**	**27.1** / 0.82 / **0.23**
Ours w/o Opt	O	16.1 / 0.30 / 0.53	18.8 / 0.45 / 0.39	19.4 / 0.39 / 0.50	18.1 / 0.38 / 0.48	17.2 / 0.54 / 0.46	16.2 / 0.47 / 0.47	18.1 / 0.50 / 0.36	16.4 / 0.54 / 0.43	17.0 / 0.51 / 0.43
Ours w/o GO	O	21.1 / 0.48 / 0.47	23.2 / 0.65 / 0.30	23.0 / 0.53 / 0.46	22.5 / 0.55 / 0.41	27.5 / 0.85 / 0.26	23.6 / 0.72 / 0.33	25.5 / 0.78 / 0.23	25.9 / 0.76 / 0.3	25.4 / 0.78 / 0.28
Ours w. RADAM	O	21.6 / 0.48 / 0.46	23.4 / 0.66 / 0.30	23.5 / 0.56 / 0.45	22.8 / 0.57 / 0.40	27.5 / 0.85 / 0.26	24.1 / 0.74 / 0.31	25.8 / 0.80 / 0.22	26.8 / 0.80 / 0.26	26.0 / 0.80 / 0.26

Table 1. **Quantitative Comparison on mip-NeRF 360 Dataset.** "-" denotes a missing implementation. The best and second best-performing algorithms for each metric are bolded and underlineed.

Method	Bicycle		Garden		Stump		Room		Counter		Kitchen		Bonsai		Mean	
	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓
MobileNeRF [5]	201	1435 / 6638	135	723 / 340	164	1069 / 497	220	903 / 421	170	864 / 399	95	552/ 256	84	606 / 284	153	879 / 1262
NeRF2Mesh [50]	129	815 / 858	117	650 / 779	123	**415 / 820**	69	328 / 462	79	517 / 384	89	601 / 434	70	319 / 636	97	387 / 667
BakedSDF [55]	1087	111248 / 44384	513	10773M / 21072	943	19667 / 38467	385	7227 / 14110	585	12212 / 23892	659	13832 / 27056	641	13629 / 26679	687	1564M / 27951
Ours	**83**	**270 / 529**	**50**	**113 / 218**	**120**	425 / 847	**45**	**124 / 238**	**58**	**138 / 268**	**40**	**70 / 136**	**42**	**233 / 459**	**62**	**196 / 385**

Table 2. **Disk Storage (MB), Number of Vertices (V) and Faces (F).** The unit for the number of vertices and faces is 10^3. "M" denotes the 10^6 unit. We save 73× the number of triangles and reduce 11× disk storage.

alizing the wireframe mesh from BakedSDF lacks meaningful interpretation due to its extremely high face density. MobileNeRF generates a "triangle soup" that inaccurately represents the scene's geometry. Also, compared to the mesh reconstructed by NeRF2Mesh, our method shows sharp details of complex geometry and smooth surfaces on the ground or wall. Moreover, in the wireframe mesh, our method employs more triangles in intricate shapes like the bicycle or the garden vase while allocating smaller triangles to planar regions or simpler structures. While the geometry of BakedSDF exhibits superior quality compared to our mesh reconstruction, achieving this high-quality mesh necessitates approximately 73× more triangles in Table 2.

Our method effectively achieves the optimal tradeoff between rendering quality and disk storage cost. For comparison, we report two tables for rendering quality and disk storage. Table 1 reports the rendering quality using the standard PSNR, SSIM [53], and LPIPS [57] on mip-NeRF 360 dataset. Instant-NGP [39], and mip-NeRF 360 [2] are volumetric rendering approaches that do not reconstruct explicit geometry, a topic that lies outside the primary focus of our study. In Table 2, we report the disk storage in megabytes and the number of vertices and faces. For a fair comparison, all the mesh files are saved in .obj and .mtl formats. Texture images are stored as .png files. Other assets, such as view-dependency MLP, are saved as .json files, except BakedSDF, which only provides gLTF format files. Our method demonstrates comparable rendering quality in outdoor scenes (with a 0.3 PSNR drop) and indoor scenes (with a 0.1 PSNR increase) while saving 73× triangles and 11× storage compared to the state-of-the-art BakedSDF. In comparison to NeRF2Mesh and MobileNeRF, our method shows better-rendering quality (Table 1), demands less disk storage, and fewer triangles (Table 2), and captures superior mesh quality (Fig. 4). In Table 5, we measure the rendering speed and GPU memory by running in-browser on a MacBookPro (2020, M1).

Method	Mesh	Deep Blending [18]		Tanks & Temples [25]	
		DrJohnson	Playroom	Barn	Courthouse
Instant-NGP [39]	X	27.7 / 0.84 / 0.38	19.5 / 0.78 / 0.46	23.4 / 0.72 / 0.43	20.0 / 0.68 / 0.53
mip-NeRF360 [2]	X	29.1 / 0.90 / 0.24	29.6 / 0.90 / 0.25	27.2 / 0.81 / 0.32	21.3 / 0.72 / 0.47
MobileNeRF [5]	O	25.9 / 0.78 / **0.31**	28.7 / 0.85 / **0.28**	24.4 / 0.77 / 0.24	17.7 / 0.54 0.36
NeRF2Mesh [50]	O	24.0 / 0.75 / 0.51	26.6 / 0.83 / 0.41	23.4 / 0.72 / 0.43	18.2 / 0.63 / 0.56
Ours	O	**26.5** / **0.82** / 0.40	**29.1** / **0.87** / 0.32	**26.5** / **0.83** / **0.24**	**20.2** / **0.73** / **0.34**

Table 3. **Quantitative Comparison.** We report PSNR ↑, SSIM ↑, and LPIPS ↓ on Deep Blending and Tanks & Temples datasets.

Method	DrJohnson		Playroom		Barn		Courthouse	
	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓	Disk ↓	V ↓ / F ↓
MobileNeRF [5]	174.9	602 / 285	281.8	1097 / 521	289.2	1080 / 502	263.6	934 / 417
NeRF2Mesh [50]	108.6	414 / 815	118.7	401 / 793	131.9	397 / 793	145.7	451 / 843
Ours	**26.7**	**46 / 92**	**60.9**	**203 / 401**	**40**	**86 / 171**	**47.4**	**97 / 192**

Table 4. **Disk Storage (MB), Vertices (V) and Faces (F).**

FPS / GPU (MB)	MobileNeRF	NeRF2Mesh	Ours
MacBookPro (1920x1080 Resolution)	59 / 324	205 / 130	212 / 115

Table 5. **FPS and GPU memory** on mip-NeRF 360 dataset

Figure 5 compares our method to the state-of-the-art baselines that focus on rendering quality: NeRF2Mesh [50], BakedSDF* [55], and MobileNeRF [5]. In their first stage, BakedSDF* represents the volumetric rendering results obtained from a surface-based volumetric representation. The visual results of our approach exhibit sharper details than other methods. In the Bicycle and Room scenes, the mesh surface of the armrest for a bench and tree leaves exhibits noticeable jiggling, resulting in distorted images. Unlike BakedSDF*, our method effectively manages the thin structure of the wheel.

Experimental Results on Deep Blending and Tank & Temples Datasets: We train and evaluate our method on four scenes: Playroom and DrJohnson from the Deep Blending dataset, and Barn and Courthouse from the Tanks & Temples dataset. The former includes bounded indoor scenes, while the latter comprises large unbounded outdoor scenes. Their capture methodology differs from that of the mip-NeRF 360 dataset, and they do not utilize 360-degree scenes. In Tanks & Temples, we exclude sky regions during training and evaluation, as mesh-based rendering tends to

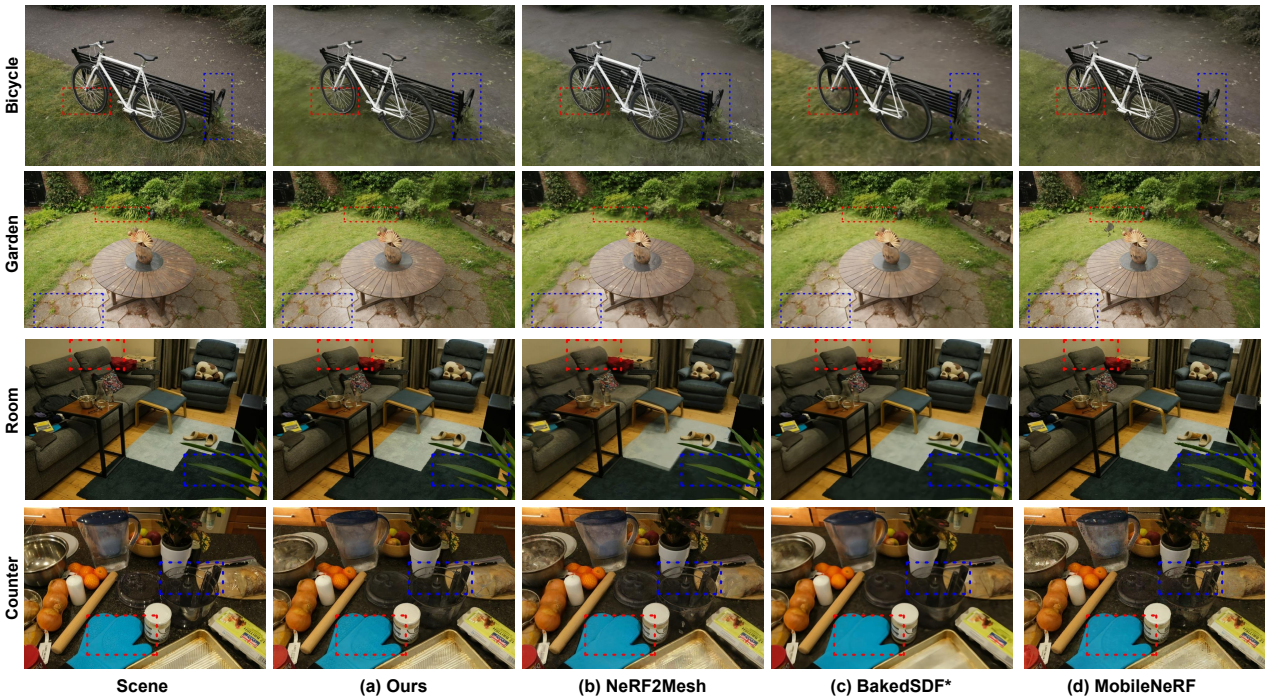

Figure 5. **Qualitative Comparisons with Existing Methods.** We visually compare our proposed technique with NeRF2Mesh [50], BakedSDF* [55], and MobileNeRF [5]. The visual results of BakedSDF* generated in its first stage through volumetric rendering. Red and blue dotted lines emphasize subtle differences in rendering quality.

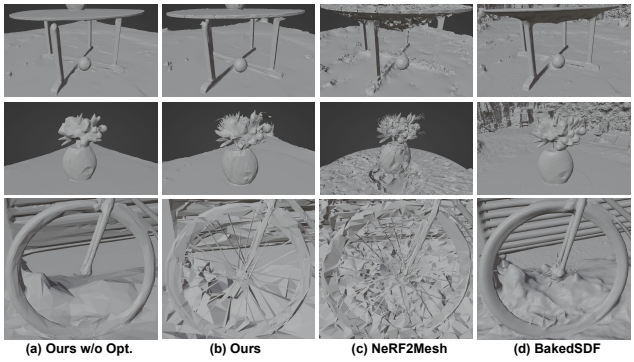

Figure 6. **Zoom-in Shading Mesh** to see more geometric details.

encounter challenges with far-away sky regions.

Ablation: In Fig. 6, we visualize the zoom-in shading mesh extracted without texture. "Ours w/o Opt" denotes we did not execute our optimization stage in Section 3.3 after mesh processing. The optimization stage enables us to represent the geometric details of garden vases and bicycle wheels. Our method can represent more detailed structures than BakedSDF and a smoother surface than NeRF2Mesh. Moreover, it is noteworthy that when employing rasterization-based rendering with the initial color model without the optimization stage, the rendering quality is poor, as shown in Table 1. In this table, "Ours w/o Geo" only trains appearance model f_{color} without training vertex deformation. Training vertex deformation yields a 1.7 gain in indoor scenes by refining the mesh surface. "Ours w. RADAM" denotes that we use the rotation-invariant

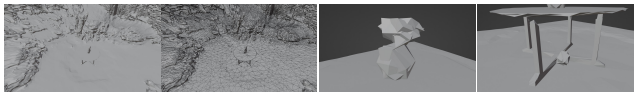

Figure 7. **Naive QSlim Results**

ADAM (RADAM) for updating mesh vertices instead of a naive ADAM. However, it provides benefits for preventing mesh distortion. In Fig. 7, we show the visualization results of naive QSlim [12]. The naive QSlim leads to shrinkage and degeneration during the edge collapse process, and the boundary is over-smoothed compared to our method.

5. Conclusion, Limitations, and Future Work

This paper introduces a practical approach for reconstructing geometrically accurate meshes with photorealistic rendering quality. Our method achieves a compact mesh representation in unbounded large-scale scenes while preserving highly detailed geometry. Furthermore, the appearance field corresponding to this lightweight mesh is compatible with the classical rasterization pipeline, enabling efficient rendering. However, our method has limitations. BakedSDF employs a well-designed appearance model based on spherical Gaussians. The potential for enhancing rendering quality exists by incorporating a more sophisticated appearance model. Moreover, the gradient-based mesh optimization in the final stage may still result in mesh distortions, including self-intersections. We plan to employ a carefully designed optimizer to mitigate such distortions in future work.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 1, 2

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 1, 2, 3, 4, 5, 6, 7

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2

[4] 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. *Advances in neural information processing systems*, 32, 2019. 3

[5] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 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*, pages 16569–16578, 2023. 1, 2, 3, 6, 7, 8

[6] Jaehoon Choi, Dongki Jung, Taejae Lee, Sangwook Kim, Youngdong Jung, Dinesh Manocha, and Donghwan Lee. Tmo: Textured mesh acquisition of objects with a mobile device by using differentiable rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16674–16684, 2023. 2

[7] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 115–122, 1998. 2

[8] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *Acm siggraph computer graphics*, 22(4):21–30, 1988. 2

[9] Carl Erikson and Dinesh Manocha. Gaps: General and automatic polygonal simplification. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 79–88, 1999. 2

[10] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2

[11] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. *Advances In Neural Information Processing Systems*, 33:9936–9947, 2020. 2, 3

[12] Michael Garland. *Quadric-based polygonal surface simplification*. Carnegie Mellon University, 1999. 2, 4, 5, 8

[13] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. 2, 4

[14] Michael Garland and Paul S Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 263–269. IEEE, 1998. 2

[15] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 4

[16] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising. *Advances in Neural Information Processing Systems*, 35:22856–22869, 2022. 2, 3, 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, 3

[18] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 6, 7

[19] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 2, 3

[20] Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198, 1997. 2

[21] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org. 3

[22] Justin Johnson, Nikhila Ravi, Jeremy Reizenstein, David Novotny, Shubham Tulsiani, Christoph Lassner, and Steve Branson. Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses*, pages 1–1. 2020. 3

[23] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 3

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

[25] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 6, 7

[26] Leif Kobbelt. Sqrt(3)-subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112, 2000. 6

[27] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for

high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 2, 3, 5

[28] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018. 2, 3

[29] Peter Lindstrom. Out-of-core simplification of large polygonal models. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 259–262, 2000. 3

[30] Peter Lindstrom and Claudio T Silva. A memory insensitive technique for large model simplification. In *Proceedings Visualization, 2001. VIS'01.*, pages 121–550. IEEE, 2001. 3

[31] Selena Zihan Ling, Nicholas Sharp, and Alec Jacobson. Vectoradam for rotation equivariant geometry optimization. *Advances in Neural Information Processing Systems*, 35:4111–4122, 2022. 3

[32] Jeffrey Yunfan Liu, Yun Chen, Ze Yang, Jingkang Wang, Sivabalan Manivasagam, and Raquel Urtasun. Real-time neural rasterization for large scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8416–8427, 2023. 2

[33] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019. 3

[34] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 1, 3, 4

[35] Fan Lu, Yan Xu, Guang Chen, Hongsheng Li, Kwan-Yee Lin, and Changjun Jiang. Urban radiance field representation with deformable neural mesh primitives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 465–476, 2023. 2

[36] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable monte carlo rendering. In *Computer Graphics Forum*, volume 40, pages 101–113. Wiley Online Library, 2021. 5

[37] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 3

[38] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 3

[39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 1, 2, 3, 7

[40] 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

[41] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021. 2, 3

[42] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021. 1, 2

[43] Werner Palfinger. Continuous remeshing for inverse rendering. *Computer Animation and Virtual Worlds*, 33(5):e2101, 2022. 2, 3, 5, 6

[44] Marie-Julie Rakotosaona, Fabian Manhardt, Diego Martin Arroyo, Michael Niemeyer, Abhijit Kundu, and Federico Tombari. Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2023. 1, 2, 6, 7

[45] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 2

[46] Jarek Rossignac and Paul Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Modeling in computer graphics: methods and applications*, pages 455–465. Springer, 1993. 2, 3

[47] William J Schroeder, Jonathan A Zarge, and William E Lorensen. Decimation of triangle meshes. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 65–70, 1992. 2

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

[49] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023. 2, 3

[50] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. *arXiv preprint arXiv:2303.02091*, 2022. 1, 2, 3, 5, 6, 7, 8

[51] Chao Wang and Xiaohu Guo. Plane-based optimization of geometry and texture for rgb-d reconstruction of indoor scenes. In *2018 International Conference on 3D Vision (3DV)*, pages 533–541. IEEE, 2018. 3

[52] 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. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 1, 2

[53] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7

[54] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neu-*

ral Information Processing Systems, 34:4805–4815, 2021. 1, 2, 4

[55] 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*, 2023. 1, 2, 4, 6, 7, 8

[56] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022. 6

[57] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7