QUADify: Extracting Meshes with Pixel-level Details and Materials from Images

Maximilian Frühauf[1,2] Hayko Riemenschneider[1] Markus Gross[2] Christopher Schroers[1]
[1]ETH Zürich [2]DisneyResearch|Studios
{max.fruehauf, hayko, christopher.schroers}@disneyresearch.com grossm@inf.ethz.ch

Abstract

Despite exciting progress in automatic 3D reconstruction from images, excessive and irregular triangular faces in the resulting meshes still constitute a significant challenge when it comes to adoption in practical artist workflows. Therefore, we propose a method to extract regular quad-dominant meshes from posed images. More specifically, we generate a high-quality 3D model through decomposition into an easily editable quad-dominant mesh with pixel-level details such as displacement, materials, and lighting. To enable end-to-end learning of shape and quad topology, we QUADify a neural implicit representation using our novel differentiable re-meshing objective. Distinct from previous work, our method exploits artifact-free Catmull-Clark subdivision combined with vertex displacement to extract pixel-level details linked to the base geometry. Finally, we apply differentiable rendering techniques for material and lighting decomposition to optimize for image reconstruction. Our experiments show the benefits of end-to-end re-meshing and that our method yields state-of-the-art geometric accuracy while providing lightweight meshes with displacements and textures that are directly compatible with professional renderers and game engines.

1. Introduction

Creating 3D content for entertainment platforms is both time-consuming and entails technical and artistic skills. Industry and academia have recently been working on automatic 3D content creation to reduce production costs and allow artists to prioritize creativity. Photogrammetry [42, 49], is a classical solution for converting multiple images of a real-world object into a 3D model. This multi-step process typically consists of stages such as multi-view stereo, feature extraction & matching, geometric extraction, texture parametrization, material baking, and de-lighting [45, 46]. Each stage has conflicting optimization objectives, introducing cumulative errors. As a result, artists spend significant amount of time manually cleaning up these 3D scans for production use.

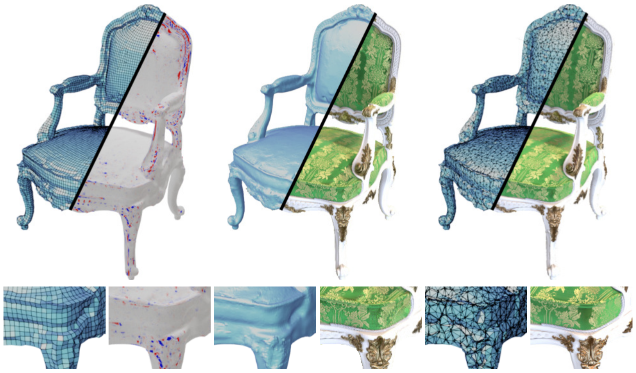


Figure 1. Our end-to-end learning uses re-meshing, subdivision, and displacement to produce a regular quad-dominant mesh, which is aligned to the surface features with pixel-level details.

In this work, we frame the task as an *inverse rendering* problem and end-to-end optimize for the surface, material, and lighting parameters which best reconstruct the captured images. While recent differentiable rendering approaches can effectively reconstruct objects, they combine geometry, material, and lighting into one neural network [26, 33, 55]. Disentangling these surface parameters has been proposed on implicit surfaces [3, 4, 60], but surface extraction usually relies on iso-surfacing techniques like Marching Cubes (MC) [29, 38], resulting in geometric inconsistencies, particularly at coarser resolutions. Working directly on triangles meshes [20, 35] is promising, but these meshes can contain "sliver" triangles, which create artifacts if the mesh deforms during animation and simulation.

Quad re-meshing aims to extract superior meshes by optimizing for regular topology aligned to a triangle mesh [2, 21, 22]. However, this can introduce inaccuracies when approximating the input mesh based on surface heuristics alone. Importantly, these quad meshes can be subdivided without introducing artifacts using Catmull-Clark subdivision [9, 17] and offer advantages in texturing and mesh editing. Existing techniques are however non-differentiable, preventing the correction of re-meshing errors by tweaking the input surface.

We present a novel method to directly extract quad-dominant meshes with decomposed materials and lighting from multi-view images of objects. We assume the object is lit under a single unknown environment with corresponding camera poses and foreground masks. Our approach extracts quad meshes by iteratively optimizing an orientation and position field, commonly used in re-meshing [2, 22, 44], while fitting the object's shape with a Signed Distance Function (SDF). We introduce a differentiable formulation of re-meshing a triangle mesh extracted from the SDF to a quad-dominant mesh by jointly learning orientation and position fields while reconstructing the surface. Further, we apply a differentiable version of Catmull-Clark subdivision and pixel-level displacement to recover details smaller than individual quad faces. Building on Nvdiffrec [35], we extract spatially-varying materials and environment lighting, producing high-quality meshes directly compatible with current production pipelines. We optimize the surface, orientation and position fields, material, and lighting parameters end-to-end and achieve competitive results in view interpolation and state-of-the-art surface accuracy with significantly improved topology driven mainly by an image loss. Our contributions include:

- An end-to-end differentiable reconstruction of images to quad-dominant meshes. We represent large-scale shapes as the surface mesh and high-frequency details as displacement and material roughness.
- A novel self-learning of orientation and position fields, enabling stable re-meshing and gradient propagation.
- An unprecedented level of mesh details through pixel-level displacement of the quad mesh using differentiable Catmull-Clark subdivision.

2. Related Work

Classical methods for Multi-View 3D reconstruction traditionally find image correspondences to estimate depth maps or combine shapes in a voxel grid. For depth-based methods, a point cloud is reconstructed by identifying correspondences between images, limited by the difficulty of finding accurate point matches [46]. Voxel-based methods, on the other hand, optimize over a voxel grid by enforcing photometric multi-view consistency [5], but are limited by the cubic memory requirement with increasing resolution.

Neural implicit methods have recently gained in popularity for novel view synthesis [33, 34] and reconstruction [26, 55] tasks. NeRF [33] learns a volumetric density field similar to the voxel-based approach described above but fails to extract high-quality surfaces due to a lack of surface constraints. NeuS [55, 56] and Neuralangelo [26], instead learn a Signed Distance Field (SDF), enabling high-quality surface reconstructions but cannot be used directly in traditional rendering pipelines, which require an explicit mesh. Even though a mesh can be extracted at inference time, this introduces additional errors not accounted for in the optimization [47], resulting in either excessive geometry to represent the scene or limited accuracy at low resolutions.

Method	Geometry	Edit.	Diff.	Fact.	Details
NeRF [33, 34]	NeurVol		✓		Implicit
Neuralangelo [26]	NeurSurf		✓		Implicit
NeuS [55, 56]	NeurSurf		✓		Implicit
Nvdiffrec [20, 35]	Tri		✓	✓	Normal map
FlexiCubes [48]	Tri		✓	✓	Normal map
InstantMeshes [22]	Quad	✓			
Our	Quad	✓	✓	✓	Displacement

Table 1. Taxonomy of methods: Geometry can be a neural volume / surface, triangle or quad-dominant mesh. Checkmarks inform if the mesh is easily editable, the method is fully differentiable, can factorize materials & lighting, and finally how details are realized.

Explicit methods commonly extract a 3D mesh from images by optimizing a reconstruction loss on a fixed mesh topology [10, 30, 54, 57]. Recent work explores differentiable meshing of an implicit field [11, 12, 27, 47, 48], allowing for changing topology during optimization. Even though extracting an explicit mesh from an SDF is generally not globally continuous, in practice, local discontinuities do not obstruct the optimization when using modern momentum-based optimizers [23, 47]. In particular, Nvdiffrec [20, 35] extracts a triangle mesh using Deep Marching Tetrahedra (DMTet) [47]. However, their meshes contain many skinny, "sliver" triangles, making them unsuitable for further processing as observed by concurrent work [48]. In this work we directly extract regular quad meshes.

Field-aligned quad re-meshing optimizes a smoothly varying orientation field on the surface to extract a quad-dominant mesh from a triangle mesh [2]. Paired with a position field aligned to these orientations prescribing the placement of vertices and face connectivity, regularly sized quad faces are extracted [21, 22, 31, 41]. InstantMeshes [22] proposes a purely local optimization of these fields on a triangle mesh, while QuadriFlow [21] considers a more global formulation, improving the re-meshing quality at the cost of runtime. However, the quad-dominant meshes extracted by these methods only approximate the input triangle mesh. In contrast, our work enables differentiating through the re-meshing process, and allows for compensating these approximation errors by adapting the triangle mesh and optimal topology alignment based on the input images. Tab. 1 shows a comparison of these methods.

3. Preliminaries

Neural SDF. Using a Signed Distance Field (SDF) to represent an optimizable surface is a common choice, as the surface can implicitly be represented by its zero-level set $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid s(\mathbf{x}) = 0\}$. For each position $\mathbf{x}$ the SDF

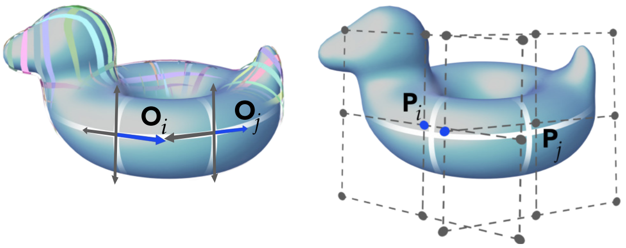


Figure 2. The orientation field (left) defines the dominant direction $\mathbf{o}_i, \mathbf{o}_j$ within the symmetry group of each vertex. The position field (right) contains lattice points $\mathbf{p}_i, \mathbf{p}_j$ aligned to $\mathbf{o}_i, \mathbf{o}_j$.

$s(\mathbf{x})$ measures the signed distance to the surface, positive outside and negative inside. Learning such an SDF $s(\mathbf{x})$ has shown to be tractable when parameterized by a Multi-Layer Perceptron (MLP) [26, 55, 56].

Multi-resolution hash grid encoding. Combining an MLP with a multi-resolution hash-grid encoding to learn $s(\mathbf{x})$ proves to be effective both in terms of representational power and memory consumption [26, 34]. For each vertex $\mathbf{x}$, the hash-grid encoder $\text{enc}(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}^{f\times d}$ linearly interpolates a feature vector $\mathbf{F}_i \in \mathbb{R}^d$ from a grid at each level of the hierarchy. Instead of representing dense feature grids in memory, a spatial hash maps query positions to features, which get concatenated together to form the final feature vector $\mathbf{F} \in \mathbb{R}^{f\times d}$. For details, we refer the reader to [34].

Triangle mesh extraction. A mesh is usually extracted with methods such as Marching Cubes (MC) or Marching Tetrahedra (MT) by querying the SDF at the vertices of a discrete voxel grid and linearly approximating the surface location [16, 29]. For any two grid vertices $\mathbf{x}_i, \mathbf{x}_j$ with $\text{sign}(s(\mathbf{x}_i)) \neq \text{sign}(s(\mathbf{x}_j))$ on a shared edge of a cube or tetrahedron, the surface vertex $\mathbf{x}_{ij}$ is computed as:

$$\mathbf{x}_{ij} = \frac{\mathbf{x}_i s(\mathbf{x}_j) - \mathbf{x}_j s(\mathbf{x}_i)}{s(\mathbf{x}_j) - s(\mathbf{x}_i)} \tag{1}$$

Even though the above formulation has a singularity for $s(\mathbf{x}_i) = s(\mathbf{x}_j)$, Wang et al. [47] experimentally show that this case does not occur during optimization.

Orientation field. Quad-dominant re-meshing as defined by Jakob et al. [22] first computes a smooth direction field over the triangle mesh surface. Since a quad face is invariant under 90° rotations, orientations need to exhibit the same symmetry shown in Fig. 2. For each vertex $\mathbf{v}_i \in \mathcal{V}$ in the triangular mesh, its representative orientation $\mathbf{o}_i \in \mathbb{R}^3$ defines a symmetry group $\mathcal{R}(\mathbf{o}_i)$ of four rotations of $\mathbf{o}_i$ using the matrix $\text{rot}(\mathbf{n}_i, \alpha) \in \mathbb{R}^{3\times 3}$ around the vertex normal $\mathbf{n}_i$:

$$\mathcal{R}(\mathbf{o}_i) = \{\text{rot}(\mathbf{n}_i, \alpha)\mathbf{o}_i \mid \alpha \in \{0, \pi/2, \pi, 3\pi/2\}\} \tag{2}$$

Two orientations $\mathbf{o}_i, \mathbf{o}_j$ connected by an edge $(i, j) \in \mathcal{E}$ are considered smooth if they align within the symmetry group: $\arg\min_{\mathbf{o},\mathbf{k}} \measuredangle(\mathcal{R}_{k_{ij}}(\mathbf{o}_i), \mathcal{R}_{k_{ji}}(\mathbf{o}_j))^2$. Where the ambiguity introduced by the symmetry group $\mathcal{R}(\mathbf{o}_i)$ is explicitly represented by the integer variables $[k_{ij}, k_{ji}, \dots] = \mathbf{k} \in [0, 3]^{2\mathcal{E}}$, which are chosen to minimize the angle of orientations in $\mathcal{R}(\mathbf{o}_i), \mathcal{R}(\mathbf{o}_j)$ respectively. Importantly, Jakob et al. [22] show that this minimization can be performed explicitly by locally smoothing the orientations on all vertices $\mathbf{v}_i$:

$$\mathbf{o}'_i \leftarrow \mathbf{o}'_i + \mathcal{R}(\mathbf{o}_j, k_{ij}) \qquad \mathbf{o}_i \leftarrow \mathbf{o}'_i / \|\mathbf{o}'_i\| \tag{3}$$

By updating all orientations $\mathbf{o}_i$ in parallel, the smoothed orientation field $\mathbf{o}^* \in \mathbb{R}^{|\mathcal{V}|\times 3}$ can be computed efficiently.

Position field. Given a smoothed orientation field and fixed edge length $s \in \mathbb{R}$, an isometric quadrangulation of the surface can be extracted by aligning a lattice with origin $\mathbf{p}_i \in \mathbb{R}^3$ at each vertex $\mathbf{v}_i$ in direction $\mathbf{o}_i$, shown in Fig. 2. Similar to the orientation field, the position field is symmetric under integer translations of length s in directions $\mathbf{o}_i$ and $\mathbf{o}_i \times \mathbf{n}_i$. Precisely, all translations $\mathbf{t} \in \mathbb{Z}^2$ are symmetric:

$$\mathcal{T}(\mathbf{p}, \mathbf{n}, \mathbf{o}, \mathbf{t}) = \mathbf{p} + s \begin{bmatrix} \mathbf{o} & \mathbf{o} \times \mathbf{n} \end{bmatrix} \mathbf{t} \tag{4}$$

Where $\mathbf{t}$ explicitly encodes the lattice point in the symmetry group $\mathcal{T}(\mathbf{p}, \mathbf{n}, \mathbf{o}, \mathbf{t})$. Jakob et al. [22] consider adjacent lattices as smooth if they align in at least a single point. By computing compatible lattice points $\mathbf{t}_{ij}, \mathbf{t}_{ji}$ for all edges $(i, j) \in \mathcal{E}$, each vertex is explicitly smoothed:

$$\mathbf{p}'_i \leftarrow \frac{\mathcal{T}(\mathbf{p}'_i, \mathbf{n}_i, \mathbf{o}_i, \mathbf{t}_{ij}) w_i + \mathcal{T}(\mathbf{p}_j, \mathbf{n}_j, \mathbf{o}_j, \mathbf{t}_{ji})}{w_i + 1} \tag{5}$$

Where w_i is incremented each time $\mathbf{p}'_i$ is visited. $\mathbf{p}'_i$ is then rounded to the nearest lattice point to $\mathbf{p}_i$ after processing all edges [22]. Finally, the smoothed positions $\mathbf{p}^* \in \mathbb{R}^{|\mathcal{V}|\times 3}$ define consistent points of the local lattice around each vertex as shown in Fig. 2. Since each lattice is aligned to the smoothed orientation field, the quad-dominant mesh extracted from $\mathbf{p}^*$ smoothly aligns to the surface.

4. Approach

We present an end-to-end method to extract quad-dominant meshes from multi-view images that allows extracting pixel-level details using displacements. Our approach is summarized in Fig. 3. Using posed images with foreground masks as supervision, our method directly optimizes for a high-quality quad-dominant mesh while simultaneously extracting decomposed materials and lighting. Hence, unlike related work in inverse rendering, we directly produce simple, regular geometry suitable for subdivision and displacement. Our meshes can be easily edited by artists using traditional modeling tools such as Blender [13] shown in the supplementary material.

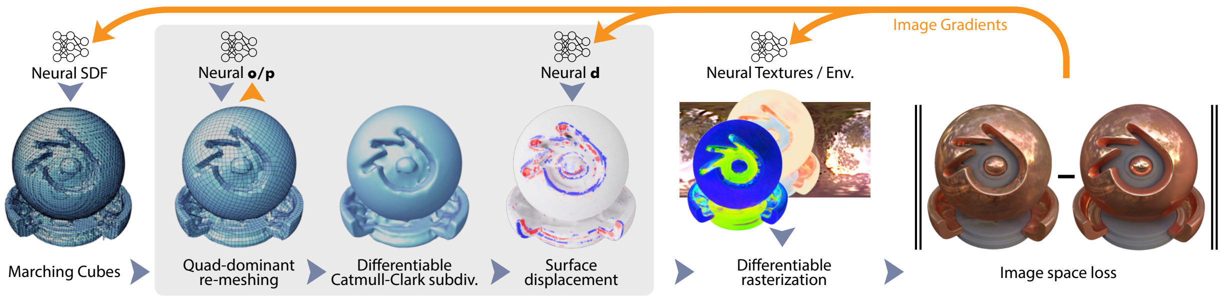


Figure 3. **Overview of our approach.** We jointly learn decomposed geometry, materials, and lighting using only posed 2D images as supervision. From a neural SDF, we extract a triangle mesh using differentiable Marching Cubes. By self-learning orientation and position fields $\mathbf{o}/\mathbf{p}$, we re-mesh the triangle mesh into a quad-dominant one and subdivide it with Catmull-Clark subdivision. On top of this subdivided mesh, we extract pixel-level details using a neural displacement field. Finally, we use a differentiable material and lighting model and back-propagate the image space loss. Our key contributions are the end-to-end learning of quad-dominant topology enabling subdivision and pixel-level displacement, highlighted in gray. 3D model from Eduardo Maldonado and env. probe by Greg Zaal (CC0).

4.1. Self-learning of Orientation and Position Fields

The key idea of our approach is a novel combination of re-meshing of a triangle mesh and inverse rendering: We develop a self-learning formulation to obtain an orientation and position field at each iteration and use these fields to differentiably extract a quad-dominant mesh. We represent the surface using an MLP-based SDF varying smoothly between iterations which implies that the derived optimal quad-dominant topology also varies smoothly. Therefore, the smoothed fields are also similar between iterations permitting self-learning them.

Optimizing the SDF jointly with the orientation and position fields has two advantages: Firstly, since the SDF is constantly changing during optimization, so is the topology of the triangle input mesh extracted using Marching Cubes. As the explicit field smoothing in Eq. 3 and Eq. 5 relies on the local topology of the triangle mesh to smooth the orientation and position fields, jointly learning the SDF implicitly regularizes this smoothing. This prevents the local smoothing iterations from getting stuck in local minima, as noted by Jakob et al. [22]. Secondly, re-meshing slightly changes the surface location of the triangle input mesh by placing freely rotated quad faces on the surface. This error can be compensated by the SDF, as shown in Fig. 4.

In the following, we introduce our self-learning reformulation of field-aligned re-meshing [22]. By using a world-space $\mathbf{o}$-MLP and $\mathbf{p}$-MLP to learn the fields, we jointly optimize surface and topology.

Orientation field self-learning. At each iteration, the $\mathbf{o}$-MLP predicts the initial value $\hat{\mathbf{o}}_i$ for the orientation smoothing in Eq. 3 at each vertex $\mathbf{v}_i$ of the triangle mesh:

$$\hat{\mathbf{o}}_i \leftarrow \mathbf{o}\text{-MLP}(\mathbf{v}_i) \quad \hat{\mathbf{o}}_i \leftarrow \frac{\hat{\mathbf{o}}_i - \mathbf{n}_i \langle \hat{\mathbf{o}}_i, \mathbf{n}_i \rangle}{\| \hat{\mathbf{o}}_i - \mathbf{n}_i \langle \hat{\mathbf{o}}_i, \mathbf{n}_i \rangle \|} \tag{6}$$

Since $\hat{\mathbf{o}}_i \in \mathbb{R}^3$ is a normalized vector in the tangent plane of $\mathbf{v}_i$, it only has one degree-of-freedom. However, we found a full 3D representation of $\hat{\mathbf{o}}_i$ could be more effectively learned by the $\mathbf{o}$-MLP. To supervise this network, we explicitly consider the $\pi/2$-symmetry of the orientation field in the loss, since all integer rotations $\mathcal{R}(\mathbf{o})$ around normal $\mathbf{n}_i$ represent the same quad face orientation. Therefore, we base our self-learning loss on the van-Mises distribution $1 - \exp(\cos\theta - 1)$ [32] with an increased winding frequency similar to Dielen et al. [15]:

$$\mathcal{L}_o(\hat{\mathbf{o}}, \mathbf{o}^*) = 1 - \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \exp(\cos(4\theta_i) - 1) \tag{7}$$

Where θ_i is the angle between $\hat{\mathbf{o}}_i$ and $\mathbf{o}_i^*$ which is minimized within the symmetry group. $\mathcal{L}_\text{o}$ can be computed efficiently as shown in the supplementary material.

Position field self-learning. We use a similar $\mathbf{p}$-MLP to predict an initial position offset for each input vertex:

$$\hat{\mathbf{p}}_i \leftarrow \tanh(\mathbf{p}\text{-MLP}(\mathbf{v}_i)) \quad \hat{\mathbf{p}}_i \leftarrow \mathbf{v}_i + s\mathbf{T}_i \hat{\mathbf{p}}_i \in \mathbb{R}^3 \tag{8}$$

Where the 2D offset $\hat{\mathbf{p}}_i$ is projected to the tangent plane of $\mathbf{v}_i$ using the projection matrix $\mathbf{T}_i \in \mathbb{R}^{3\times2}$, scaled by the re-meshing length s, and used as the initial value for position smoothing. Importantly, $\mathbf{T}_i$ is independent of $\mathbf{o}_i$, decoupling self-learning of $\hat{\mathbf{p}}_i$ from the orientation $\mathbf{o}_i^*$. Measuring the deviation of the predicted position $\hat{\mathbf{p}}_i$ and the smoothed one $\mathbf{p}_i^*$ is done in tangent space since both are two degree-of-freedom quantities. However, the two tangent spaces of $\hat{\mathbf{p}}_i$ and $\mathbf{p}_i$ generally do not have the same basis due to the projection $\mathbf{T}_i$. So we project both to the lattice aligned with $\mathbf{o}_i^*$ before measuring the deviation:

$$\mathcal{L}_p(\hat{\mathbf{p}}, \mathbf{p}^*) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \left\| \frac{1}{s} \begin{bmatrix} {\mathbf{o}_i^*}^T \\ (\mathbf{o}_i^* \times \mathbf{n}_i)^T \end{bmatrix} (\hat{\mathbf{p}}_i - \mathbf{p}_i^*) \right\|^2 \tag{9}$$

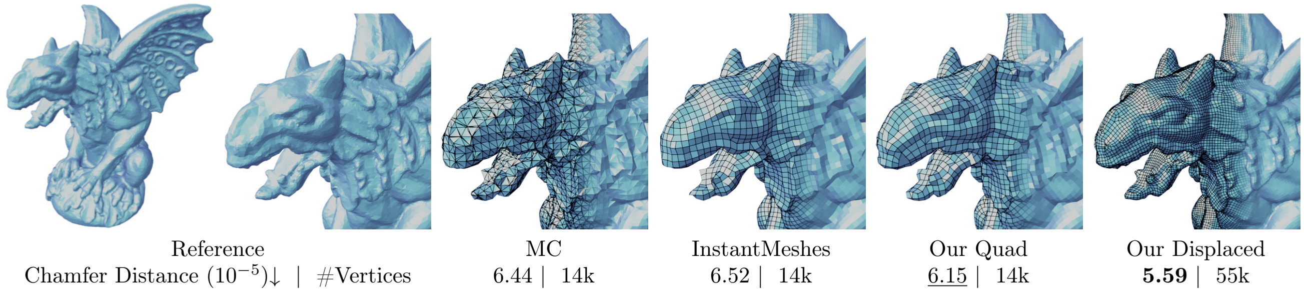


Figure 4. Mesh extraction from depth and mask images via SDF supervision by the reference mesh [36]. We extract meshes at matching vertex counts across methods and report Chamfer distance wrt. the reference mesh. Differentiable Marching Cubes (MC) [29] reconstructs the surface but fails to align topology to surface features. Re-meshing the MC mesh using InstantMeshes [22] as a post-processing step aligns quad faces to the surface but is not differentiable and, therefore, cannot account for surface errors introduced by re-meshing. Our method also extracts a quad-dominant mesh while end-to-end optimizing the underlying SDF for reconstruction accuracy. Fine details can be extracted by differentiably subdividing and displacing our quad-dominant mesh. Gargoyle model is from Myles et al. [36].

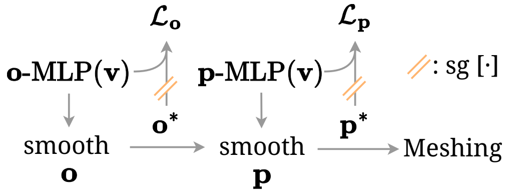


Figure 5. The self-learning losses $\mathcal{L}_o$ and $\mathcal{L}_p$ supervise the orientation and position field, respectively, by treating $\mathbf{o}^*$ and $\mathbf{p}^*$ as constants derived by smoothing from the predicted initial fields.

Combined loss. To learn both fields jointly, we combine both losses in the $\mathcal{L}_{op}$ loss on the smoothed fields:

$$\mathcal{L}_{op} = \mathcal{L}_o(\hat{\mathbf{o}}, \text{sg}\,[\mathbf{o}^*]) + \mathcal{L}_p(\hat{\mathbf{p}}, \text{sg}\,[\mathbf{p}^*]) \tag{10}$$

By using the stop-gradient operation sg $[\cdot]$ on the smoothed fields $\mathbf{o}^*, \mathbf{p}^*$, we enforce the MLPs to self-learn the fields from their predictions $\hat{\mathbf{o}}, \hat{\mathbf{p}}$, as illustrated in Fig. 5.

Differentiation at the optimum. Since we re-mesh the Marching Cubes mesh at each iteration, gradients need to be back-propagated from the quad-dominant mesh to the triangle input mesh, in turn supervising the SDF. However, the orientation and position fields can change drastically during field smoothing, causing unwanted gradients. As we run multiple explicit smoothing steps in Eq. 3 and Eq. 5 each iteration, the gradients of the smoothing process fail to supervise the SDF, as shown in Sec. 5.4.

To still propagate meaningful gradients through the field smoothing and supervise the SDF, even if self-learning has not yet converged, we skip these smoothing gradients. Intuitively, by assuming the inputs to the field smoothing are already optimal, i.e. $\hat{\mathbf{o}} = \mathbf{o}^*$ and $\hat{\mathbf{p}} = \mathbf{p}^*$, gradients of the smoothing process can be removed. We enforce this is assumption by copying gradients on the smoothed fields to the initial values similar to Vector Quantization [53]:

$$\mathbf{o}^* \leftarrow \hat{\mathbf{o}} + \text{sg}\,[\mathbf{o}^* - \hat{\mathbf{o}}] \qquad \mathbf{p}^* \leftarrow \hat{\mathbf{p}} + \text{sg}\,[\mathbf{p}^* - \hat{\mathbf{p}}] \tag{11}$$

During forward computation, the above equation evaluates to the identity: $\mathbf{o}^* \leftarrow \mathbf{o}^*$, $\mathbf{p}^* \leftarrow \mathbf{p}^*$, while the gradients of the smoothed fields get assigned to the respective initial values skipping the field smoothing. Doing so allows for learning the surface location jointly with field smoothing.

Quad-dominant face extraction. After orientation and position field smoothing, we extract a quad-dominant mesh from the position field by collapsing edges referring to the same lattice points, following Jakob et al. [22]. Since collapsing edges is a discrete process, we do not track gradients w.r.t the clustering but only the merged quad vertices.

4.2. Pixel-level Subdivision & Displacement

Inspired by production renderers [8], we represent small-scale details below the re-meshing edge length s using Catmull-Clark subdivision [9, 17] and vector displacement, as shown in Fig. 3. In contrast to previous methods, we extract pixel-level details by displacing the learned geometry and do not rely on shading tricks such as normal maps.

Following the algorithm by Dupuy et al [17], we iteratively subdivide the quad-dominant mesh while tracking gradients. Since subdivision only refines the re-meshed surface and not the voxel grid, we can efficiently subdivide each face until it reaches the size of individual pixels in the reference image (typically after 2 subdivision steps). Small-scale details are extracted by jointly learning a displacement field on the subdivision surface:

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \boldsymbol{d}(\mathbf{v}_i) \qquad \boldsymbol{d}(\mathbf{v}_i) = \frac{s}{2}\tanh(\boldsymbol{d}\text{-MLP}(\mathbf{v}_i)) \tag{12}$$

Each vertex is perturbed by the displacement $\boldsymbol{d}(\mathbf{v}_i) \in \mathbb{R}^3$, chosen such that only details smaller than the quad faces are extracted by displacement, as shown in Fig. 6. This implicitly decomposes the surface into a low-frequency mesh with a high-frequency displacement.

4.3. Optimization & Implementation

Losses. Let θ be our optimization parameters, including SDF, $\mathbf{o}, \mathbf{p}$ fields, materials and light probe. Given reference images I_i^{ref} with camera pose $T_i \in \mathbb{R}^{4\times 4}$ we minimize the empirical risk by rendering an image $I^\theta(T)$ using a differentiable renderer [25]: $\arg\min_\theta \mathbb{E}_T[\mathcal{L}_{\text{total}}(I^\theta(T), I^{\text{ref}}(T))]$. Where $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{mask}} + \lambda_{op}\mathcal{L}_{\text{op}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}$ are an image space loss ($\|\cdot\|_1$ on tonemapped colors), mask loss ($\|\cdot\|_2^2$) the field loss and regularizers as used by Nvdiffrec [35]. We give details in the supplementary material.

Materials and Lighting. We use the physically-based (PBR) material model by Disney [6] commonly used in differentiable rendering [19, 20, 59]. We adopt the shading model of Nvdiffrec [35] and represent materials with a diffuse $\boldsymbol{k}_{\text{d}}$ and specular term $\boldsymbol{k}_{\text{orm}}$ learned by a world-space MLP. Current rendering pipelines store materials as 2D textures on the surface, requiring a 2D parametrization on the mesh [8]. This is incompatible with the world-space texturing used during optimization. Therefore, we follow Nvdiffrec and re-parametrize the model once the surface and materials have converged. We generate unique texture coordinates using Blender [13], initialize 2D textures ($\boldsymbol{k}_{\text{d}}, \boldsymbol{k}_{\text{orm}}, \boldsymbol{d}$) from the world-space MLPs and continue optimizing using frozen topology. Details in the supplement.

Progressive level of detail. During optimization, we iteratively enable the levels of our SDF and material hash encoding, effectively regularizing the surface reconstruction by first fitting low-frequency surface features and then learning high-frequency ones [26, 56, 58].

Enabling the finest subdivision level throughout the surface optimization causes the surface to get stuck in local minima, especially in concave regions. Since each subdivided vertex is the weighted average of neighboring vertices in the next coarser subdivision level, gradients on the vertices of the finer subdivision level contribute to multiple vertices in the coarser level. So, vertex gradients at the coarse level are an average of the gradients on the fine level. The resulting coarse-level average is not meaningful since the gradient of neighboring vertices are substantially different if the reconstruction error is large. We sidestep this limitation by progressively enabling subdivision levels and displacement only once the surface has converged.

5. Experiments

This section evaluates our method on various applications. First, we compare to state-of-the-art mesh extraction methods in view interpolation and surface accuracy. To emphasize that our method is a drop-in replacement for current iso-surfacing approaches, we evaluate the quality of extracted meshes through differentiable iso-surfacing methods: Marching Cubes (MC) [29], DMTet [47], and the concurrent work FlexiCubes [48]. Then we compare our quad-dominant meshes to DMTet for artifact-free subdivision and displacement. Finally, we ablate our $\mathbf{o}/\mathbf{p}$ self-learning.

	View Interpolation			Geometry	
	PSNR↑	SSIM↑	LPIPS↓	vis. CD↓(10^{-2})	#V (10^3)↓
Nvdiffrec	**29.06**	**0.939**	**0.078**	7.8	40
Nvdiffrecmc	26.56	0.913	0.115	5.5	42
FlexiCubes	28.95	0.936	0.082	3.3	52
Our Quad I Disp.	27.96	0.933	0.083	3.2 I **3.0**	**32** I 516

Table 2. Mesh rendering quality metrics averaged over all 8 scenes in the NeRF Synthetic dataset [33]. We re-trained Nvdiffrec [35], NvdiffrecMC [20], and FlexiCubes [48] using public source code. Our scores are competitive while enabling edible meshes.

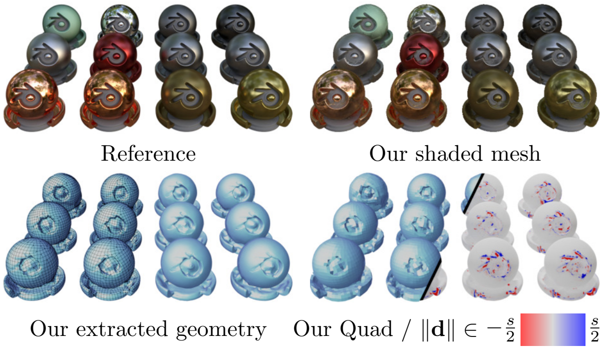


Figure 6. Mesh extraction results on the MATERIALS scene reconstructed from 100 images from the NeRF Synthetic dataset [33].

5.1. Inverse Rendering of Quad-dominant Meshes

Synthetic datasets. We show view interpolation and geometric results on the NeRF Synthetic dataset [33] in Tab. 2 and show a qualitative example of the MATERIALS scene in Fig. 6. Per-scene results are included in the supplementary material. Even though view interpolation is not the main focus of our method, it performs consistently on par with Nvdiffrec in PSNR. Perceptually-based metrics (SSIM and LPIPS) are even closer between all methods since slight inaccuracies in the silhouette of an object are less detrimental to these metrics. Comparing visible Chamfer Distance (vis. CD) highlights the advantage of aligning quad faces with the object surface: The quad-dominant mesh (Our Quad) improves reconstruction accuracy compared to related methods while using fewer vertices. Adding two subdivision levels and displacement (Our Disp.) allows our method to recover fine details, decreasing vis. CD even further. Notably, the increased vertex count due to subdivision and displacement does not hinder mesh editability, as it can be efficiently computed during rendering [40, 52].

We continuously deform the MLP SDF initialized to a sphere to fit the target shape as opposed to random initialization, which allows for re-using the orientation and po-

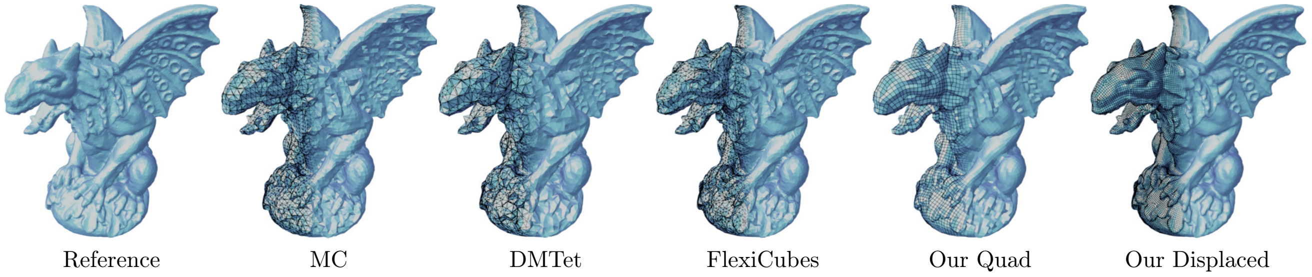


Figure 7. Visual comparison of different mesh extraction techniques. Our displaced and quad-dominant meshes, Marching Cubes (MC), DMTet [47] and FlexiCubes [48] are generated by directly supervising predicted geometry with depth and mask losses on the ground truth. Our quad-dominant meshes show clear alignment of the quad faces to object features and extraction of surface details visible in the displaced surfaces.

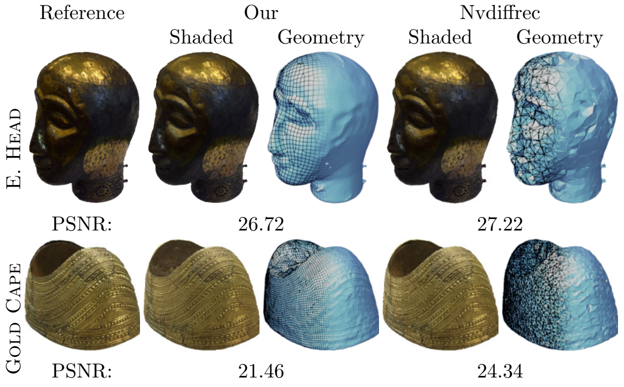


Figure 8. Reconstruction on the real-world dataset from NeRD [3], comparing our results with Nvdiffrec. Results for Nvdiffrec were re-trained from public source code. Our method extracts *regular* base geometry while recovering more fine details than Nvdiffrec, such as the groves in the GOLD CAPE scene as displacement.

sition fields between iterations. Yet, when supervised with rasterization [25] instead of volume rendering [26, 55, 56], the optimization can get stuck in local minima in poorly observed regions causing local inconsistencies. These artifacts are visible e.g. in the FICUS scene and get picked up by visual metrics evaluating the full-size rendered images.

Real-world datasets. We provide reconstructions on two scenes from the NeRD dataset [3] captured at diverse viewpoints with automatically generated (inaccurate) masks. We show qualitative view interpolation results in Fig. 8 and visually compare the extracted geometry. Meshes extracted by our method are much smoother than the ones by Nvdiffrec, since we parametrize the SDF with an MLP instead of per vertex. Additionally, we progressively enable details as described in Sec. 4.3 while constraining small-scale details to this smooth surface through displacement.

View interpolation scores are similar between both methods, but our method extracts more fine detail in geometry. However, on the GOLD CAPE scene, our method cannot recover the hole in this crown-like object due to our SDF parametrization continuously fitting a sphere to the observations. This causes a $\tilde{3}$dB drop in PSNR averaged over all views. However, the reconstruction is still accurate from frontal views, and we attribute this failure to inconsistent foreground separation of this hole in the dataset [3, 35].

Method	CD(10^{-5})↓	F1↑	AR>4(%)↓	RR>4(%)↓	#V(10^3)
MC	5.22	0.66	12.02	12.01	10.34
DMTet	5.23	0.67	16.89	16.32	10.92
FlexiCubes	4.87	0.69	6.69	8.26	11.87
Our Quad	4.96	0.67	**0.00**	**0.24**	10.66
Our Disp.	**4.63**	**0.70**	0.22	0.88	42.75

Table 3. Quantitative results on mesh reconstruction on 79 objects from the Myles dataset [36]. We report the following metrics: Chamfer Distance (CD), F1 Score, outlier percentage of Aspect Ratio (AR) & Radius Ratio (RR), and number of vertices (#V).

5.2. Mesh Reconstruction

To compare the quality of meshes extracted with our method without artifacts introduced by inverse rendering, we follow the idealized experiment setup of Shen et al. [48]: The reconstruction is supervised by rendering depth and mask images and directly supervising the SDF at 1000 points randomly sampled from the reference mesh. Details on the training setup are in the supplementary material. We select the same 79 shapes [48] from the dataset collected by Myles et al. [36] consisting of diverse shapes ranging from 3D scans to CAD models.

We show results in Tab. 3 and a visual example in Fig. 7. Compared to DMTet [47], Marching Cubes (MC) [29], and the concurrent work FlexiCubes (FC) [48] our method is the only one extracting quad-dominant meshes. Since our quadrangular faces are much more regular than the triangle ones extracted with either related work, our method almost eliminates faces with Aspect Ratio (AR) and Radius Ratio (RR) above the commonly accepted threshold of 4, shown in Tab. 3 and Fig. 9. Meanwhile, the Chamfer Distance (CD) of the quad-dominant mesh (Our Quad) is consistently

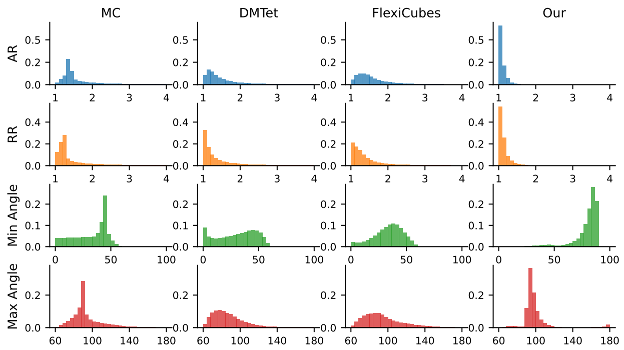


Figure 9. Qualitative comparison of Angle Ratio, Radius Ratio, and minimal and maximal face angles on all 79 objects. Our method extracts clean editable meshes with significantly more equilateral faces, resulting in lower AR and RR (rows 1 and 2) and face angles concentrated around 90 degrees (rows 3 and 4).

on par and surpasses all baselines when subdividing it once and adding displacement (Our Disp).

Reconstruction accuracy strongly depends on the mesh resolution. Therefore, to fairly compare methods, we match the number of vertices between methods by varying the SDF grid resolution, following Huang et al. [48].

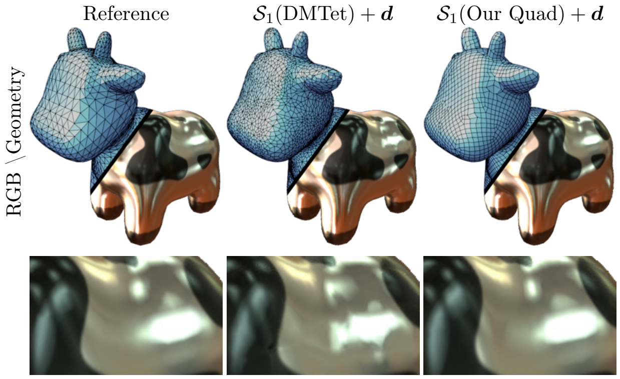
5.3. Displacement on Triangle vs. Quad Meshes

In Fig. 10, we compare subdividing and displacing a triangle mesh from DMTet with our Quad-dominant mesh. Subdividing the triangle DMTet mesh with Catmull-Clark subdivision creates irregularly sized faces due to skinny triangles visible as shading artifacts. Displacing the subdivision surface removes some artifacts but fails to accurately reconstruct the reflected highlights of the reference. Our quad-dominant topology does not suffer from these artifacts, resulting in smoother shading and improved reconstruction.

5.4. Ablations

Self-learning. In Fig. 11, we ablate the field self-learning introduced in Sec. 4.1. Learning the orientation and position fields with $\mathcal{L}_{op}$ is crucial to reconstructing the reference shape and aligning topology to surface features. Without $\mathcal{L}_{op}$ the optimization cannot progress from its spherical initialization due to both fields changing significantly each iteration when re-initialized randomly. This causes the surface location to flicker and therefore fails to back-propagate consistent gradients to the SDF.

Gradient skipping. At each iteration, we perform six explicit smoothing steps of the orientation and position field but do not track gradients for the smoothing iterations. In Fig. 11, we show the necessity of skipping gradients during smoothing, as the surface shape cannot be reconstructed without it. We experimentally found skipping smoothing gradients to be an effective solution for convergence issues.


Reference
$\mathcal{S}_1(\text{DMTet}) + \boldsymbol{d}$
$\mathcal{S}_1(\text{Our Quad}) + \boldsymbol{d}$
RGB \ Geometry

PSNR↑ | SSIM↑	<u>32.29</u> | <u>0.9825</u>	**32.79** | **0.9854**
CD (10^{-5})↓	<u>3.93</u>	**3.89**

Figure 10. Visual artifacts from subdividing and displacing DMTet vs. our quad-dominant mesh. Both are trained on a vast amount of 2D observations (40k views) with known lighting. We optimize the base DMTet and our quad mesh for 1000 steps each, then apply one subdivision level $\mathcal{S}_1$ and displacement $\boldsymbol{d}$, then run another 4000 steps. View interpolation metrics are the mean of 100 test images. Spot by Keenan Crane [14].

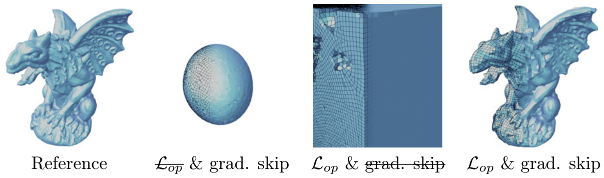

Reference
~~$\mathcal{L}_{op}$~~ & grad. skip
$\mathcal{L}_{op}$ & ~~grad. skip~~
$\mathcal{L}_{op}$ & grad. skip

Figure 11. Ablation results. Self-learning the orientation and position fields with $\mathcal{L}_{op}$ is required to reconstruct the object and faces to align to surface features. Gradient skipping is essential to propagating meaningful gradients to the SDF.

6. Conclusion and Future Work

In summary, we propose a method to extract coarse quad-dominant meshes with pixel-level displacement from images by end-to-end optimizing the appearance of the rendered mesh. Our results surpass state-of-the-art in geometric accuracy while reconstructing much more regular meshes, enabling artifact-free subdivision and displacement to extract pixel-level details previously only possible at excessive voxel resolutions. By design, our method directly extracts objects compatible with rendering and game engines, enabling a multitude of applications and significantly simplifying workflows by automatically extracting meshes closer to those created by human artists. Re-meshing covers the triangle mesh with coarser quad-faces, decreasing the geometry resolution compared to the input triangle mesh, and requires a continuous surface throughout the optimization. Therefore, we do not use a discrete per-vertex SDF but a continuous MLP-based one, which is more prone to getting stuck in local minima, especially in concave regions. In future work, we hope to address this limitation and improve the reconstruction of challenging scenes.

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 *ICCV*, pages 5835–5844, 2021. 2

[2] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-Integer Quadrangulation. *ACM Trans. Graph.*, 28, 2009. 1, 2

[3] 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*, pages 12664–12674, 2021. 1, 7

[4] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P.A. Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *NeurIPS*, 2021. 1

[5] A. Broadhurst, T.W. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *ICCV*, pages 388–393, 2001. 2

[6] Brent Burley. Physically Based Shading at Disney. *SIGGRAPH Talk*, 2012. 6

[7] Brent Burley and Dylan Lacewell. Ptex: Per-Face Texture Mapping for Production Rendering. *Computer Graphics Forum*, 27(4):1155–1164, 2008. 1

[8] Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. The Design and Evolution of Disney's Hyperion Renderer. *ACM Trans. Graph.*, 37(3):1–22, 2018. 5, 6, 1

[9] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978. 1, 5

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

[11] Zhiqin Chen and Hao Zhang. Neural Marching Cubes. *ACM Trans. Graph.*, 40(6):1–15, 2021. 2

[12] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Trans. Graph.*, 41 (4):104:1–104:13, 2022. 2

[13] Blender Online Community. *Blender - a 3D Modelling and Rendering Package*, 2018. 3, 6, 1, 2, 7

[14] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Robust fairing via conformal curvature flow. *ACM Trans. Graph.*, 32(4):1–10, 2013. 8

[15] Alexander Dielen, Isaak Lim, Max Lyon, and Leif Kobbelt. Learning Direction Fields for Quad Mesh Generation. *Computer Graphics Forum*, 40(5):181–191, 2021. 4

[16] Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Transactions on Information and Systems*, 74:214–224, 1991. 3

[17] J. Dupuy and K. Vanhoey. A Halfedge Refinement Rule for Parallel Catmull-Clark Subdivision. *Computer Graphics Forum*, 40(8):57–70, 2021. 1, 5, 7

[18] Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *CVPR*, pages 2463–2471, 2017. 2

[19] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-Driven Automatic 3D Model Simplification. In *EGSR*, 2021. 6

[20] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising. In *NeurIPS*, pages 22856–22869, 2022. 1, 2, 6, 3

[21] Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J. Guibas. QuadriFlow: A Scalable and Robust Method for Quadrangulation. *Computer Graphics Forum*, 37(5):147–160, 2018. 1, 2

[22] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6):1–15, 2015. 1, 2, 3, 4, 5

[23] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 2

[24] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4), 2017. 4, 5, 7

[25] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. In *ACM Trans. Graph.*, 2020. 6, 7, 4

[26] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Ming-Yu Liu, Chen-Hsuan Lin, and Mathias Unberath. Neuralangelo: High-Fidelity Neural Surface Reconstruction. *CVPR*, 2023. 1, 2, 3, 6, 7

[27] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep Marching Cubes: Learning Explicit Surface Representations. In *CVPR*, 2018. 2

[28] Pascal Lienhardt. Topological models for boundary representation: A comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82, 1991. 7

[29] William Lorensen and Harvey Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH*, 21, 1987. 1, 3, 5, 6, 7, 2

[30] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. *Computer Graphics Forum*, 40, 2021. 2

[31] Max Lyon, Marcel Campen, David Bommes, and Leif Kobbelt. Parametrization quantization with free boundaries for trimmed quad meshing. *ACM Trans. Graph.*, 38(4):1–14, 2019. 2

[32] K.V. Mardia and P.E. Jupp. *Directional Statistics*. 2009. 4

[33] 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, 6, 7

[34] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):1–15, 2022. 2, 3

[35] Jacob Munkberg, Wenzheng Chen, Jon Hasselgren, Alex Evans, Tianchang Shen, Thomas Muller, Jun Gao, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *CVPR*, 2022. 1, 2, 6, 7, 3, 5

[36] Ashish Myles, Nico Pietroni, and Denis Zorin. Robust field-aligned global parametrization. *ACM Trans. Graph.*, 33(4): 1–14, 2014. 5, 7, 2, 4

[37] Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. TorchMetrics - measuring reproducibility in PyTorch, 2022. 2

[38] G.M. Nielson. On marching cubes. *IEEE Trans. Visual. Comput. Graphics*, 9(3):283–297, 2003. 1

[39] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations Without 3D Supervision. In *CVPR*, 2020. 2, 7

[40] Matthias Nießner and Charles Loop. Analytic displacement mapping using hardware tessellation. *ACM Trans. Graph.*, 32(3):1–9, 2013. 6

[41] Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. Reliable feature-line driven quad-remeshing. *ACM Trans. Graph.*, 40(4):1–17, 2021. 2

[42] Marc Pollefeys and Luc Van Gool. From images to 3D models. *Commun. ACM*, 45:50–55, 2002. 1

[43] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, 2021. 1

[44] Nico Schertler, Marco Tarini, Wenzel Jakob, Misha Kazhdan, Stefan Gumhold, and Daniele Panozzo. Field-aligned online surface reconstruction. *ACM Trans. Graph.*, 36(4): 1–13, 2017. 2

[45] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016. 1

[46] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise View Selection for Unstructured Multi-View Stereo. In *ECCV*, 2016. 1, 2

[47] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep Marching Tetrahedra: A Hybrid Representation for High-Resolution 3D Shape Synthesis. In *NeurIPS*, pages 6087–6101, 2021. 2, 3, 6, 7, 9

[48] 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 Trans. Graph.*, 42(4):37:1–37:16, 2023. 2, 6, 7, 8, 3

[49] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006. 1

[50] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A Standard Default Color Space for the Internet - sRGB, 1996. 5

[51] C. Bane Sullivan and Alexander Kaszynski. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, 2019. 3

[52] Theo Thonat, Francois Beaune, Xin Sun, Nathan Carr, and Tamy Boubekeur. Tessellation-free displacement mapping for ray tracing. *ACM Trans. Graph.*, 40(6):1–16, 2021. 6

[53] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. In *NeurIPS*, 2017. 5

[54] Thomas Walker, Octave Mariotti, Amir Vaxman, and Hakan Bilen. Explicit neural surfaces: Learning continuous geometry with deformation fields. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, 2023. 2

[55] 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. *NeurIPS*, 34, 2021. 1, 2, 3, 7

[56] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *ICCV*, 2023. 2, 3, 6, 7

[57] Markus Worchel, Rodrigo Diaz, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert. Multi-View Mesh Reconstruction with Neural Deferred Shading. In *CVPR*, 2022. 2

[58] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields. *ICLR*, 2022. 6

[59] 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 *CVPR*, 2021. 6

[60] 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 Trans. Graph.*, 40(6):1–18, 2021. 1, 5, 6

[61] Fuqiang Zhao, Yuheng Jiang, Kaixin Yao, Jiakai Zhang, Liao Wang, Haizhao Dai, Yuhui Zhong, Yingliang Zhang, Minye Wu, Lan Xu, and Jingyi Yu. Human Performance Modeling and Rendering via Neural Animated Mesh. *ACM Trans. Graph.*, 41(6):1–17, 2022. 1