

## ExMesh: EXplicit Mesh Reconstruction with Topology Adaptation

Chuanjin Fan<sup>1</sup> Lifan Wu<sup>1</sup> Wenjie Chang<sup>1</sup> Hanzhi Chang<sup>1</sup> Wenfei Yang<sup>1</sup>† Tianzhu Zhang<sup>1,2</sup>  
<sup>1</sup>University of Science and Technology of China  
<sup>2</sup>National Key Laboratory of Deep Space Exploration, Deep Space Exploration Laboratory

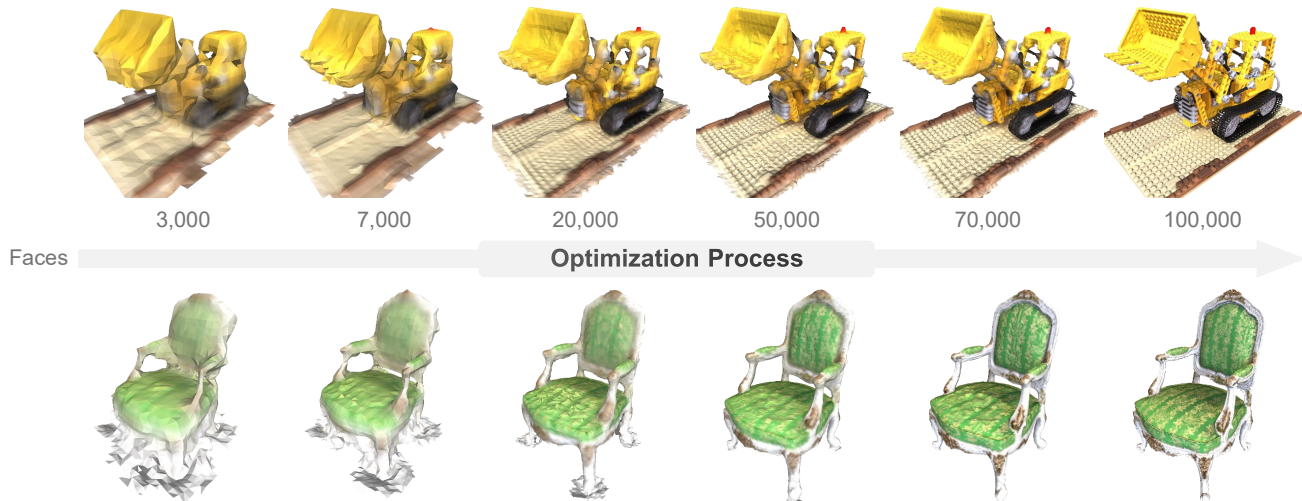


Figure 1. The coarse-to-fine optimization process of our ExMesh framework. Driven by adaptive vertex splitting and merging, together with decoupled geometry-texture optimization, our method refines a coarse mesh into a high-fidelity result. This approach achieves a superior trade-off between reconstruction quality, computational efficiency, and a concise mesh structure.

### Abstract

Reconstructing surface meshes from multi-view images has remained a core challenge in recent years. Most existing methods, whether implicit or explicit, depend on intermediate representations and post-processing steps like Marching Cubes or TSDF fusion, often resulting in artifacts and fragmented geometry. Directly optimizing explicit meshes is a promising approach. However, it presents two critical challenges. The first is how to adaptively refine mesh topology to capture detail without introducing degenerate faces. The second is how to maintain consistent UV coordinates for high-fidelity texturing as the mesh structure evolves. To overcome these, we propose ExMesh, a novel framework that directly optimizes explicit meshes by integrating differentiable optimization with discrete topology updates. Specifically, we introduce an adaptive vertex splitting and merging strategy, along with real-time UV maintenance, to enable coarse-to-fine optimization while preserving geometric integrity. To our knowledge, ExMesh is the first framework to seamlessly integrate discrete topology operations into a continuous differentiable optimization pipeline.

*Extensive experiments demonstrate that ExMesh achieves a balance among accuracy, computational efficiency, and mesh conciseness.*

### 1. Introduction

Surface meshes, as a core form of explicit geometric representation, hold irreplaceable value across various domains, including virtual/augmented reality, digital twins, and robotics. Their representation not only supports real-time rasterized rendering but can also be directly used for scene editing and interaction [1, 3, 4, 19, 44, 49]. However, reconstructing high-fidelity meshes from multi-view images faces a core challenge, which lies in balancing three aspects: accuracy, computational efficiency, and mesh conciseness.

Recently, with the development of Neural Radiance Fields, implicit methods [12, 47, 51, 57] represent scenes as continuous fields learned by MLPs. When extracting meshes, however, these methods must discretize the field into high-resolution voxels and apply Marching Cubes [29]. This process often results in blurred sharp edges, loss of thin structures, and extremely long training time. Explicit

methods using discrete Gaussian primitives (or their variants) [6, 16, 18, 23, 58, 61] have improved training and rendering efficiency. Nevertheless, they still require post-processing, such as TSDF fusion [8], to convert the unstructured Gaussians into meshes, often producing a large number of discrete fragments. Meanwhile, mesh-driven methods based on hybrid representations have emerged [13, 26, 33, 40, 41, 50, 55]. Instead of directly optimizing meshes, these methods use SDFs, point clouds, or voxels as the primary optimization targets, which increases framework complexity and can compromise geometric accuracy. Moreover, if texture is explicitly stored on the geometric carrier, its density must be increased to preserve detail, leading to a significant rise in face count [46, 47, 51, 55]. While coordinate-based MLPs [33] can decouple texture resolution from geometry, they still require complex post-training baking. These drawbacks make it difficult to maintain both high accuracy and simplicity, highlighting the necessity of moving beyond intermediate representations.

Directly optimizing position and texture is the ideal path to bypass aforementioned drawbacks [15], but it also faces severe technical challenges. At the geometry processing level, this path faces the dual challenge of how to adaptively refine the mesh and how to maintain its structural integrity. Intuitively, an effective strategy would allocate a higher density of faces to geometrically complex regions and lower density to flatter areas. However, existing global uniform refinement strategies directly perform quad-splitting on all faces [2, 11, 33, 50], resulting in significant face redundancy in flat regions and insufficient detail recovery in complex areas. On the other hand, vertex displacement during optimization can easily produce degenerate faces, leading to substantial degradation of geometric accuracy. At the texture representation level, a fundamental constraint is that the quality of texture detail is highly dependent on face density. If texture is directly stored on vertices or faces, even geometrically simple surfaces with complex textures require a massive number of faces to represent texture detail. Although using a decoupled UV map is the ideal solution, real-time maintenance of UV coordinates under changing topology remains an unsolved challenge [33, 54].

In this paper, we propose a novel mesh reconstruction framework that integrates differentiable optimization with discrete topology updates. By updating mesh structure during optimization, our method directly produces a renderable and editable mesh without intermediate representations or post-processing. Specifically, to tackle the challenges at the geometry processing level, we design an adaptive vertex splitting and merging strategy. This strategy splits vertices in complex geometric regions with high curvature and large gradients to accurately recover details. Meanwhile, it merges vertices on redundant and degenerate faces generated during optimization, which effectively controls the

face count and ensures geometric integrity. Furthermore, to break constraints at the texture representation level, we propose a method for real-time UV maintenance by updating UV coordinates alongside each topology change. By jointly optimizing vertex positions with a UV map, our method preserves texture details while significantly reducing the dependence on face count. To our knowledge, ExMesh is the first approach to achieve direct mesh optimization with real-time adaptive refinement, providing a new technical paradigm for mesh reconstruction.

In summary, our contributions are as follows:

- We propose ExMesh, a novel end-to-end mesh reconstruction framework that integrates explicit optimization with discrete topology updates, directly producing structurally complete and editable meshes.
- An adaptive vertex splitting and merging strategy that achieves coarse-to-fine mesh refinement while effectively removing redundant and degenerate faces.
- A mechanism for real-time UV maintenance, enabling decoupled geometry and texture during topology updates for high-fidelity texturing on concise meshes.

## 2. Related Work

### 2.1. Implicit Neural Surface Reconstruction

Implicit neural representations, particularly Neural Radiance Fields (NeRF) [31], have achieved impressive results in novel view synthesis and photorealistic rendering. However, the volume density field learned by NeRF makes it difficult to extract smooth and fine-grained surfaces. To address this, subsequent works use Signed Distance Functions [35, 38, 56] to implicitly represent geometry. Among these, NeuS [51], UNISURF [36] and VolSDF [57] proposed SDF-to-density transformations, enabling optimization of SDF field via a photometric loss. Recent methods further improve quality and efficiency by introducing multi-resolution hash grids [32], numerical gradients [47], and geometric priors such as SfM point clouds [12] to constrain SDF optimization. Despite advances in reconstruction quality, these methods remain indirect, relying on intermediate representations and post-processing (e.g., Marching Cubes [29]) to extract meshes, which is time-consuming and often blurs sharp edges or loses thin structures.

### 2.2. Gaussian Splatting-based Reconstruction

To overcome the high training and rendering costs of implicit methods, a series of explicit methods have been proposed. Among them, 3D Gaussian Splatting [23] is a pioneering work that uses millions of anisotropic 3DGS primitives to represent the scene. Paired with a differentiable rasterizer, it achieves fast training and real-time rendering, with rendering quality surpassing NeRF in many aspects. Subsequent works explored other explicit primitives, such

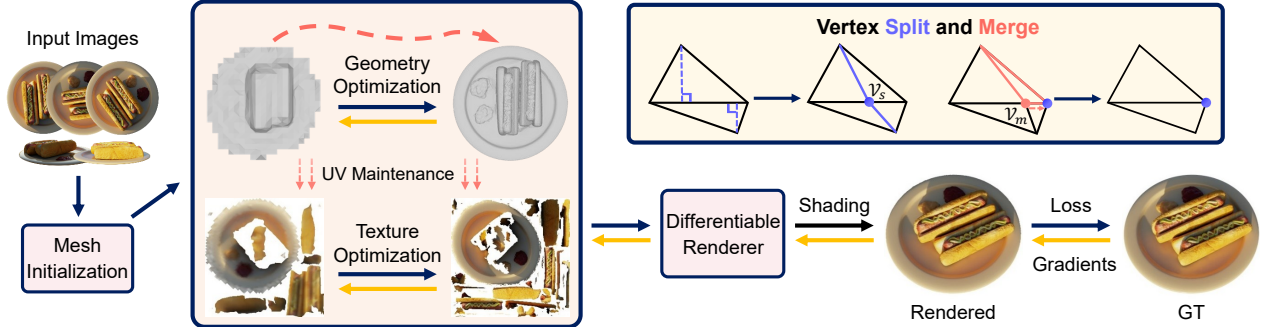


Figure 2. Overview of our ExMesh framework. We utilize a differentiable renderer to compute a photometric loss between the rendered image and the ground-truth (GT) image. The gradients are back-propagated to drive two decoupled optimization loops: (1) Geometry Optimization for vertex positions and (2) Texture Optimization for a separate UV map. The continuous geometry optimization is interleaved with discrete vertex split and merge operations to adaptively refine the mesh topology from coarse to fine.

as 2DGS [18], which flattens 3DGS into disks, and Triangle Splatting [16], which directly optimizes a “triangle soup”.

Although 3DGS and its variants have achieved great success in rendering, their primitives are essentially an unstructured point cloud, not directly editable manifold meshes. Therefore, to extract surface from these primitives, subsequent works [5, 6, 9, 18, 30, 45, 46, 53, 58, 61] also rely on post-processing steps. These methods typically produce meshes using TSDf fusion [8, 34] or Poisson reconstruction [22]. Although more detailed than Marching Cubes [29], it still results in a large number of discrete mesh fragments and noise artifacts. Whether using implicit fields or Gaussian primitives, this indirect pipeline remains a core obstacle to high-quality and concise mesh reconstruction.

### 2.3. Mesh-driven Reconstruction

To circumvent the indirect pipeline, a series of works began exploring mesh-driven reconstruction. This was enabled by the development of differentiable rasterizers [7, 21, 24, 28], which allow gradients to flow directly from image loss to mesh attributes. However, there remain two core challenges, the first of which is topology control. Early methods [21, 28] typically require a fixed topology template [13, 50], limiting the ability of geometry expressiveness. While some works explore topology changes via continuous remeshing [37] or probabilistic predictions [42, 43], they often lack adaptive, gradient-driven detail recovery. To support structural changes, later methods typically optimize intermediate carriers. Among them, Nvdiffrac [33] and Flexi-Cubes [41] optimize continuous SDF fields, while IMLS-Splatting [55] and GeoSVR [26] use point clouds or voxels. Although these methods allow topology to change during training, their optimization target remains intermediate carriers rather than explicit meshes, resulting in constrained flexibility, increased complexity, and precision bias.

The second challenge is texture representation. Directly coupling texture with the geometric carrier [26, 55] requires a much denser carrier to capture high-frequency details,

causing face count inflation. The ideal path is to use a decoupled texture map. Nvdiffrac [33] addresses this by introducing a coordinate-based MLP texture, enabling high-frequency details on low-poly meshes. However, this implicit texture lacks direct editability and requires a post-training baking step to extract a standard UV map. Furthermore, its strategy requires freezing topology mid-training to preserve mapping continuity. Consequently, existing mesh-driven methods are caught in a dilemma: either achieve texture decoupling but sacrifice adaptive topology; or achieve dynamic topology at the expense of texture decoupling.

## 3. Method

### 3.1. Overview

We propose ExMesh, a framework for reconstructing high-fidelity explicit meshes from multi-view images  $\{I\}$ . As shown in Figure 2, ExMesh takes an initial coarse mesh  $\mathcal{M}_{init}$  as input, then iteratively optimizes its geometry and texture directly. In each iteration, we use a differentiable renderer [24] to render an image  $\hat{I}$  for the current view, which is then compared with the ground-truth image  $I$ . Gradients from the loss are back-propagated to jointly optimize two components that are decoupled by design: (1) Geometry Optimization, which updates the 3D positions  $\mathbf{V}$  of the vertices; and (2) Texture Optimization, which refines a separate UV map  $\mathbf{T}$ . To enable coarse-to-fine reconstruction, we interleave continuous gradient optimization with periodic vertex splitting and merging (Sec. 3.2, Sec. 3.3), enabling dynamic mesh refinement and redundancy removal.

### 3.2. Adaptive Vertex Splitting

Our topology update strategy borrows the idea of vertex splitting from mesh generation [60], and combines it with a density control strategy from 3DGS [23], allowing geometric complexity to be added adaptively. Vertex splitting process consists of two stages: criterion and operation.

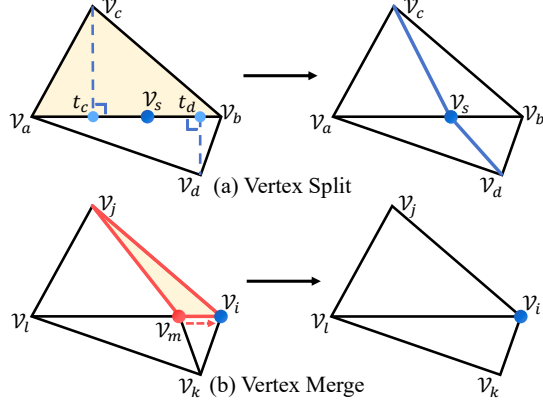


Figure 3. Our discrete topological operations. (a) Vertex Split: A new vertex  $v_s$  is inserted to split the edge  $(v_a, v_b)$ , replacing 2 faces with 4 new ones. (b) Vertex Merge: Vertex  $v_m$  is merged into  $v_i$  by collapsing the edge  $(v_i, v_m)$ , simplifying the local topology.

**Splitting Criterion.** Whether a face is worth refining depends on two perspectives: the significance of its optimization error and the complexity of its geometry. In terms of optimization dynamics, the first metric we track is the vertex EMA gradient magnitude ( $\mathcal{G}_v$ ), which uses an Exponential Moving Average (EMA) to accumulate the position gradient norm received by each vertex  $\mathbf{V}$  during optimization. A high  $\mathcal{G}_v$  indicates the vertex is located in a region requiring significant geometric adjustment. Its update formula is:

$$\mathcal{G}_v^{(t)} = (1 - \beta_g)\mathcal{G}_v^{(t-1)} + \beta_g\|\nabla_{\mathbf{V}}^{(t)}\| \quad (1)$$

where  $t$  denotes the current iteration step,  $\beta_g$  is the EMA decay factor, and  $\nabla_{\mathbf{V}}^{(t)}$  represents the gradient at the current step. For geometric priors, we evaluate curvature  $\mathcal{K}_f$ , which geometrically assesses local complexity. We estimate  $\mathcal{K}_f$  of face  $\mathcal{F}$  by calculating the average angle between its normal  $\mathbf{n}_f$  and the normals  $\mathbf{n}_{adj}$  of all adjacent faces  $\mathcal{F}_{adj}$ :

$$\mathcal{K}_f = \frac{1}{N_{adj}} \sum_{\mathcal{F}_{adj}} \arccos(\mathbf{n}_f \cdot \mathbf{n}_{adj}) \quad (2)$$

When performing splitting, we first calculate EMA of the gradient magnitude  $\mathcal{G}_f$  for each face (i.e., the mean  $\mathcal{G}_v$  of three vertices). To avoid ineffectual splitting in already detailed regions, candidate faces are limited to the top 50% by area. A weighted score  $S_f$  combining optimization needs and geometric complexity is used for sampling:

$$S_f = \alpha\mathcal{G}_f + \beta\mathcal{K}_f \quad (3)$$

where  $\alpha$  and  $\beta$  are hyperparameters to balance the weights of gradient and curvature. The sampled faces are selected as splitting targets for the current round.

**Splitting Operation.** For a selected face  $\mathcal{F}(v_a, v_b, v_c)$ , we select the edge with the highest score, computed as the ratio of edge length to vertex degree, i.e.,  $S_e = l_e/d_e$ , as the splitting edge, which jointly considers geometric and topological factors. The main challenge is to determine the position

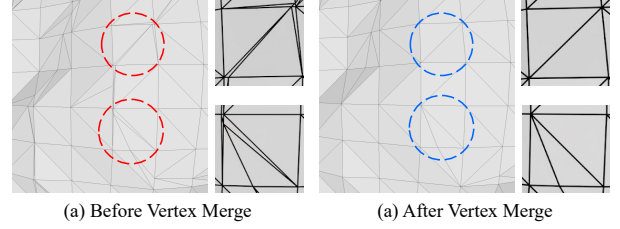


Figure 4. Visualization of our vertex merge operation, which eliminates degenerate faces to restore a clean topology.

of new vertex  $v_s$  (see Figure 3 (a)). If  $e$  is a non-boundary edge, it must be shared with another face  $\mathcal{F}'(v_a, v_b, v_d)$ . To fit local surface morphology, we project  $v_c$  and  $v_d$  onto  $e$  and set  $v_s = (t_c + t_d)/2$ , where  $t_c$  and  $t_d$  are projection points. Subsequently, topology update is performed by deleting two original faces  $\mathcal{F}$  and  $\mathcal{F}'$ , then adding four new faces  $\mathcal{F}(v_a, v_s, v_c)$ ,  $\mathcal{F}(v_s, v_b, v_c)$ ,  $\mathcal{F}(v_a, v_s, v_d)$ , and  $\mathcal{F}(v_s, v_b, v_d)$ . Conversely, if  $e$  is a boundary edge, adjacent face  $\mathcal{F}'$  does not exist. In this case, we place  $v_s$  at the midpoint of  $e$ , then delete  $\mathcal{F}$ , add two new faces  $\mathcal{F}(v_a, v_s, v_c)$  and  $\mathcal{F}(v_s, v_b, v_c)$ .

Finally, to ensure the stability of topology update, two constraints are applied. The current split operation will be skipped if: (1) the adjacent face  $\mathcal{F}'$  has already been modified in previous operations, to avoid concurrent modification conflicts; or (2) the relative position  $\alpha = \|v_s - v_a\|/\|v_b - v_a\|$  of  $v_s$  on edge  $e$  must fall within the central interval  $[0.25, 0.75]$ , to avoid introducing degenerate faces.

### 3.3. Adaptive Vertex Merging

As the counterpart to vertex splitting, the vertex merging mechanism is analogous to the classic ‘‘edge collapse’’ in mesh simplification [14, 17]. It aims to restore clean topology by removing redundant and degenerate faces.

**Merging Criterion.** Whether a face  $\mathcal{F}$  needs to be merged is judged from two perspectives: rendering visibility and geometric morphology. Rendering visibility is measured by tracking a render contribution count  $\mathcal{C}_{render}(\mathcal{F})$ , which records how many times each face  $\mathcal{F}$  contributes to the rendered output after rasterization. If  $\mathcal{C}_{render}(\mathcal{F}) = 0$  for a face, it indicates the face was never rendered during training (e.g., becoming an internal face or being completely occluded), and is thus considered redundant. For geometric morphology, to address the degenerate faces produced during optimization, we calculate each face’s degeneracy  $\mathcal{D}_f$ :

$$\mathcal{D}_f = \frac{\text{Area}(\mathcal{F})}{l_{\max}^2(\mathcal{F})} \quad (4)$$

where  $l_{\max}(\mathcal{F})$  denotes the length of the longest edge in face  $\mathcal{F}$ . If  $\mathcal{D}_f$  is less than threshold  $\tau_{degen}$ , the face is considered degenerate. Finally, only faces in the bottom 50% by area that meet either criterion are added to the merge set  $\mathcal{M}_{merge}$ , restricting the process to fine local details. Fig-

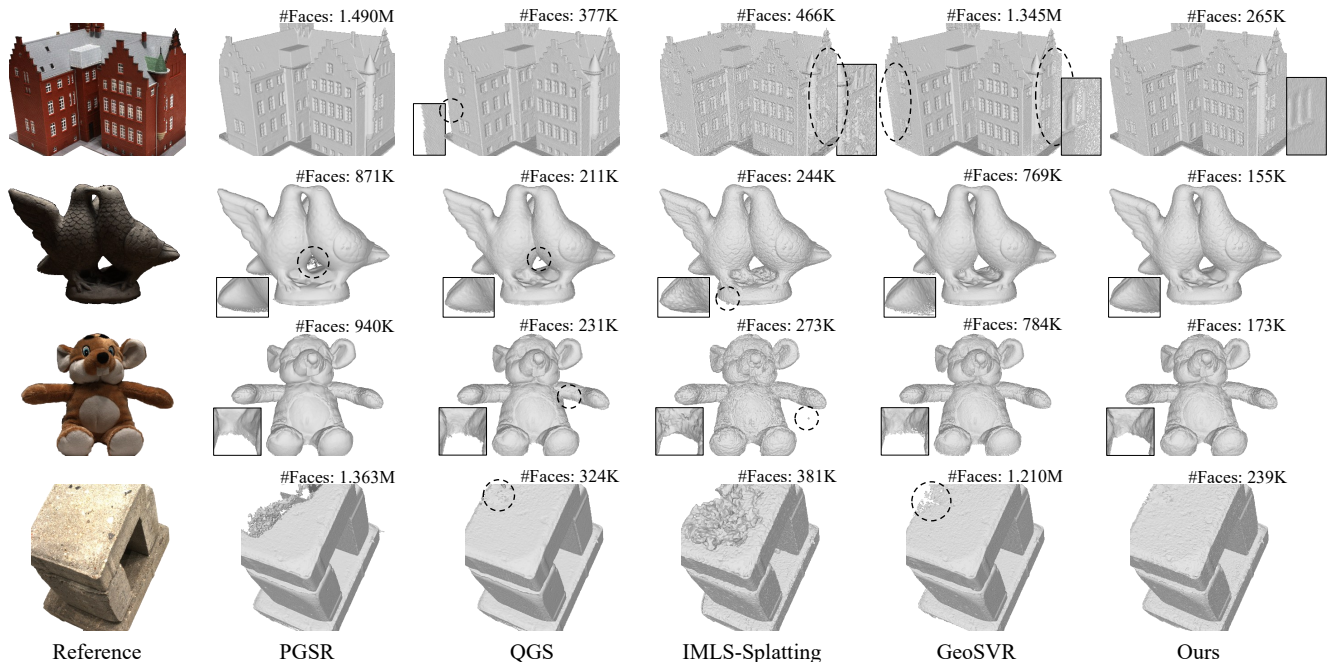


Figure 5. Qualitative geometric comparison on the DTU dataset. Our method produces structurally clean and high-fidelity meshes.

ure 4 intuitively demonstrates this process; our mechanism can identify these degenerate forms and clean them up.

**Merging Operation.** For each face  $\mathcal{F}(v_i, v_j, v_m)$  in  $\mathcal{M}_{merge}$ , we first determine the edge  $e_{collapse}$  to collapse. If  $\mathcal{F}$  is a boundary face, we collapse its boundary edge  $e_{collapse} = (v_i, v_m)$  to maintain boundary integrity. If  $\mathcal{F}$  is an interior face, we collapse edge  $e_{collapse}$  with the smallest  $S_e$ , in order to efficiently simplify the local mesh structure. After selecting  $e_{collapse}$ , we compare the vertex degrees of its two endpoints,  $v_i$  and  $v_m$ . If  $v_m$  has a lower degree, it will be merged into  $v_i$ .

The topology update operation is shown in Figure 3 (b): first, vertex  $v_m$  is deleted. For adjacent faces that contained both  $v_i$  and  $v_m$  (i.e.,  $\mathcal{F}(v_i, v_j, v_m)$  and  $\mathcal{F}(v_i, v_k, v_m)$ ), they will be degenerated into an edge, so we directly delete them. For adjacent faces that contained  $v_m$  but not  $v_i$  (e.g.,  $\mathcal{F}(v_j, v_l, v_m)$  and  $\mathcal{F}(v_l, v_k, v_m)$ ), they must be re-linked to preserve the local manifold structure. We replace  $v_m$  with  $v_i$  in these faces, updating them to  $\mathcal{F}(v_j, v_l, v_i)$  and  $\mathcal{F}(v_l, v_k, v_i)$ . Thus, each merge operation in the non-boundary case deterministically deletes the two faces and re-links the remaining adjacent faces of  $v_m$ . To avoid topological errors from concurrent modifications, a merge operation will be skipped if a face requiring modification has already been affected by previous operations.

### 3.4. Geometry and Texture Optimization

Our framework adopts a decoupled geometry and texture optimization strategy, where geometry is represented by the vertex positions  $\mathbf{V}$  of mesh  $\mathcal{M}$ , while texture is stored in a

separate, fixed-resolution UV map  $\mathbf{T}$ , with UV coordinates  $\mathbf{u}$  linking them. In the rendering pipeline, vertex positions  $\mathbf{V}$  determine the mesh projection, while interpolated UV coordinates  $\mathbf{u}$  are used to sample the UV map  $\mathbf{T}$  for surface color [33]. Image loss gradients are independently back-propagated to drive geometry and texture optimization.

**Geometry Optimization.** Geometry optimization updates all vertex positions  $\mathbf{V}$  to align the rendered mesh shape with the GT image. This is achieved by minimizing a composite loss function (see Sec. 3.5), which provides gradients that push  $\mathbf{V}$  toward the correct spatial positions when discrepancies exist between the rendered and GT images.

**Texture Optimization.** Texture optimization adjusts all texels on the UV map  $\mathbf{T}$  by back-propagating image rendering loss through the differentiable UV sampling operation, which effectively “paints” high-definition texture onto the mesh. This decoupling allows the resolution of  $\mathbf{T}$  to be independent of the face count, enabling our method to represent high-resolution texture details with a concise mesh and thus avoid face count inflation.

The texture decoupling mechanism must also coordinate with dynamic topology updates to correctly maintain UV coordinates. When a vertex split operation creates a new vertex  $v_s$ , its UV coordinate  $u_s$  should be initialized appropriately. If  $v_s$  is on a boundary edge  $e(v_a, v_b)$ , its UV coordinate  $u_s$  is set to the midpoint of  $u_a$  and  $u_b$ . For non-boundary edges,  $v_s$  is shared by faces  $\mathcal{F}(v_a, v_b, v_c)$  and  $\mathcal{F}'(v_a, v_b, v_d)$ , which may lie on different UV islands. We perform interpolation in the UV space of  $\mathcal{F}$  and  $\mathcal{F}'$  to obtain two candidate UV coordinates:  $u_s^{(1)} = (1 - \mu)u_a + \mu u_b$  and

Table 1. Quantitative geometric comparison on the DTU dataset. Our method achieves comparable accuracy to SOTA methods while using significantly fewer faces. ■ denotes the best, ■ second best, ■ third best, respectively.

| Method      |                   | 24                                       | 37                                       | 40                                       | 55                                       | 63                                       | 65                                       | 69                                       | 83                                       | 97                                       | 105                                      | 106                                      | 110                                      | 114                                      | 118                                      | 122                                      | Avg.                                     | Time                                    | Faces                                    |
|-------------|-------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|---|--|
| NeRF-based  | VolSDF [57]       | 1.14                                     | 1.26                                     | 0.81                                     | 0.49                                     | 1.25                                     | 0.70                                     | 0.72                                     | 1.29                                     | 1.18                                     | 0.70                                     | 0.66                                     | 1.08                                     | 0.42                                     | 0.61                                     | 0.55                                     | 0.86                                     | >12h                                    | 1.58M                                    |
|             | NeuS [51]         | 0.83                                     | 0.98                                     | 0.56                                     | 0.37                                     | 1.13                                     | 0.59                                     | 0.60                                     | 1.45                                     | 0.95                                     | 0.78                                     | 0.52                                     | 1.43                                     | 0.36                                     | 0.45                                     | 0.45                                     | 0.77                                     | >12h                                    | 798K                                     |
|             | Neuralangelo [47] | 0.45                                     | 0.74                                     | 0.33                                     | <span style="color: yellow;">0.34</span> | 1.05                                     | <span style="color: orange;">0.54</span> | 0.53                                     | 1.33                                     | 1.05                                     | 0.72                                     | <span style="color: orange;">0.43</span> | 0.69                                     | 0.34                                     | <span style="color: yellow;">0.38</span> | 0.42                                     | 0.62                                     | >12h                                    | 1.98M                                    |
| GS-based    | SuGaR [46]        | 1.47                                     | 1.33                                     | 1.13                                     | 0.61                                     | 2.25                                     | 1.71                                     | 1.15                                     | 1.63                                     | 1.62                                     | 1.07                                     | 0.79                                     | 2.45                                     | 0.98                                     | 0.88                                     | 0.79                                     | 1.33                                     | 1h                                      | 962K                                     |
|             | 2DGS [18]         | 0.46                                     | 0.84                                     | <span style="color: yellow;">0.31</span> | 0.45                                     | 0.92                                     | 1.01                                     | 0.83                                     | 1.23                                     | 1.30                                     | 0.66                                     | 0.61                                     | 1.07                                     | 0.45                                     | 0.71                                     | 0.54                                     | 0.76                                     | <span style="color: red;">11m</span>    | 260K                                     |
|             | TriSplat [16]     | 0.76                                     | 0.97                                     | 0.77                                     | 0.46                                     | 1.77                                     | 1.27                                     | 1.04                                     | 1.75                                     | 1.72                                     | 0.80                                     | 0.91                                     | 1.20                                     | 0.56                                     | 0.91                                     | 0.97                                     | 1.05                                     | <span style="color: yellow;">15m</span> | 265K                                     |
|             | GOF [58]          | 0.50                                     | 0.82                                     | 0.37                                     | 0.37                                     | 1.12                                     | 0.74                                     | 0.73                                     | 1.18                                     | 1.29                                     | 0.68                                     | 0.77                                     | 0.90                                     | 0.42                                     | 0.66                                     | 0.49                                     | 0.74                                     | 1h                                      | 1.06M                                    |
|             | PGSR [6]          | <span style="color: orange;">0.36</span> | <span style="color: orange;">0.57</span> | 0.38                                     | <span style="color: red;">0.33</span>    | 0.78                                     | 0.58                                     | <span style="color: yellow;">0.50</span> | <span style="color: yellow;">1.08</span> | <span style="color: orange;">0.63</span> | <span style="color: yellow;">0.59</span> | 0.46                                     | <span style="color: yellow;">0.54</span> | <span style="color: red;">0.30</span>    | <span style="color: orange;">0.38</span> | <span style="color: orange;">0.34</span> | <span style="color: orange;">0.52</span> | 30m                                     | 1.05M                                    |
|             | QGS [61]          | <span style="color: yellow;">0.38</span> | <span style="color: yellow;">0.62</span> | 0.37                                     | 0.38                                     | <span style="color: orange;">0.75</span> | <span style="color: yellow;">0.55</span> | <span style="color: yellow;">0.51</span> | 1.12                                     | <span style="color: orange;">0.68</span> | 0.61                                     | 0.46                                     | <span style="color: orange;">0.58</span> | 0.35                                     | 0.41                                     | 0.40                                     | <span style="color: yellow;">0.54</span> | 48m                                     | 252K                                     |
| Mesh-driven | Nvdiffrac [33]    | 3.04                                     | 3.02                                     | 2.10                                     | 0.78                                     | 2.18                                     | 1.60                                     | 1.46                                     | 1.67                                     | 2.85                                     | 1.26                                     | 1.10                                     | 3.26                                     | 1.13                                     | 1.31                                     | 1.19                                     | 1.86                                     | >1h                                     | <span style="color: red;">132K</span>    |
|             | Flexicubes [41]   | 1.66                                     | 1.60                                     | 0.91                                     | 0.50                                     | 2.60                                     | 1.32                                     | 0.87                                     | 1.45                                     | 1.60                                     | 1.38                                     | 0.73                                     | 1.85                                     | 1.01                                     | 0.64                                     | 0.75                                     | 1.26                                     | >1h                                     | <span style="color: orange;">168K</span> |
|             | IMLS-Splat [55]   | 0.63                                     | 1.21                                     | 0.67                                     | 0.62                                     | 1.04                                     | 0.82                                     | 0.78                                     | 1.32                                     | 1.09                                     | 0.89                                     | 0.69                                     | 0.95                                     | 0.62                                     | 0.63                                     | 0.65                                     | 0.84                                     | <span style="color: yellow;">15m</span> | 418K                                     |
|             | GeoSVR [26]       | <span style="color: red;">0.32</span>    | <span style="color: red;">0.51</span>    | <span style="color: red;">0.30</span>    | <span style="color: red;">0.33</span>    | <span style="color: red;">0.71</span>    | <span style="color: red;">0.48</span>    | <span style="color: red;">0.42</span>    | <span style="color: red;">1.03</span>    | <span style="color: red;">0.62</span>    | <span style="color: red;">0.56</span>    | <span style="color: red;">0.33</span>    | <span style="color: red;">0.46</span>    | <span style="color: red;">0.30</span>    | <span style="color: red;">0.34</span>    | <span style="color: red;">0.32</span>    | <span style="color: red;">0.47</span>    | 49m                                     | 1.12M                                    |
|             | Ours              | 0.44                                     | 0.70                                     | <span style="color: red;">0.30</span>    | <span style="color: yellow;">0.34</span> | <span style="color: yellow;">0.76</span> | 0.63                                     | 0.52                                     | <span style="color: orange;">1.06</span> | 1.05                                     | <span style="color: yellow;">0.58</span> | <span style="color: yellow;">0.45</span> | 0.88                                     | <span style="color: yellow;">0.32</span> | <span style="color: orange;">0.36</span> | <span style="color: orange;">0.38</span> | 0.58                                     | <span style="color: orange;">13m</span> | <span style="color: orange;">196K</span> |

$u_s^{(2)} = (1 - \mu)u'_a + \mu u'_b$ , where  $u'_a$  and  $u'_b$  are the UV coordinates of  $v_a$  and  $v_b$  in  $\mathcal{F}'$ , and  $\mu$  is the relative position of  $v_s$  on edge  $e$ . If  $\|u_s^{(1)} - u_s^{(2)}\| < \tau_{uv}$ , then set  $u_s$  to their average, where  $\tau_{uv}$  is the threshold. Conversely,  $v_s$  is associated with two distinct UV coordinates,  $u_s^{(1)}$  and  $u_s^{(2)}$ , which are stored independently to ensure a proper texture seam. Correspondingly, when a vertex  $v_m$  is removed in vertex merging, any unreferenced UV coordinates are deleted.

### 3.5. Training Strategy

Our model is trained using a composite loss  $\mathcal{L}$  defined as:

$$\mathcal{L} = \lambda_{rgb}\mathcal{L}_{rgb} + \lambda_d\mathcal{L}_d + \lambda_m\mathcal{L}_m + \lambda_s\mathcal{L}_s + \lambda_b\mathcal{L}_b \quad (5)$$

where  $\lambda$  is the weight coefficient for each term. Specifically,  $\mathcal{L}_{rgb}$  is the photometric loss, using the same  $(1 - \lambda_{dssim})\mathcal{L}_{L1} + \lambda_{dssim}\mathcal{L}_{D-SSIM}$  combination as 3DGS [23].  $\mathcal{L}_d$  is the Pearson depth loss, which leverages depth reference from Depth Anything 3 [27]. While  $\mathcal{L}_m$  is the silhouette loss, defined as the binary cross-entropy between predicted and ground-truth masks [21, 28]. To ensure geometric smoothness,  $\mathcal{L}_s$  applies Laplacian smoothing [10, 48]. Finally,  $\mathcal{L}_b$  is the bi-vertex offset regularization loss as defined in FlexiCubes [41].

## 4. Experiments

### 4.1. Implementation Details

Our method is implemented based on the PyTorch framework, utilizing CUDA kernels for acceleration, and using nvdiffrast [24] for differentiable rendering. All experiments were completed on a single NVIDIA RTX 3090 GPU. Our pipeline begins with an initial coarse mesh, which is obtained by training 2DGS [18] for 7k iterations (about 2 mins) and extracting it using TSDF [8] at a resolution of  $256^3$ . Before training, we perform multiple rounds of vertex merging to eliminate degenerate faces in the initial mesh.

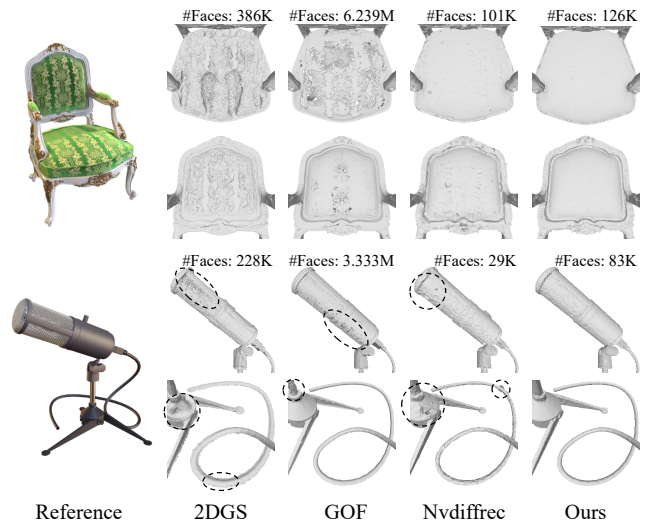


Figure 6. Qualitative comparison on the NeRF-synthetic dataset.

During optimization, firstly, a 1k-iteration warm-up is used for initial geometry and texture fitting. Then, during the 1k–7k phase, vertex splitting and merging are performed every 500 iterations and the UV coordinates are updated accordingly, with UV map reconstruction every 2000 iterations [25, 39]. Finally, in the last 1k iterations, the topology is frozen for final refinement. Detailed hyperparameters and loss weights are provided in the supplementary material.

### 4.2. Comparison

**Datasets.** Our experiments are conducted on two benchmark datasets. The DTU dataset [20] contains 15 real-world objects, each including 49 or 69 images at a resolution of  $1600 \times 1200$ . For efficiency, we downsample the images to  $800 \times 600$  resolution, as in [51, 57]. NeRF-synthetic dataset [31] contains 8 object-level scenes, each with 100 training images and 200 test images at  $800 \times 800$  resolution.

**Baselines.** We compare our method against three categories of surface reconstruction methods: (1) NeRF-based

Table 2. Quantitative comparison on the NeRF-synthetic dataset.

| Method                 | Mic  | Chair | Ship | Material | Lego | Drums | Ficus | Hotdog | Avg. | Time | Faces  |
|------------------------|------|-------|------|----------|------|-------|-------|--------|------|------|--------|
| NeuS [51]              | 0.53 | 0.37  | 1.25 | 0.35     | 0.83 | 0.86  | 0.40  | 0.96   | 0.69 | >12h |        |
| Neuralangelo [47]      | 0.52 | 0.29  | 0.34 | 0.85     | 0.61 | 0.77  | 0.54  | 0.87   | 0.60 | >12h | 8.387M |
| 2DGS <sup>+</sup> [18] | 1.39 | 0.92  | 1.92 | 1.11     | 0.75 | 0.88  | 1.35  | 0.92   | 1.16 | 11m  | 639K   |
| TriSplat               | 1.14 | 1.65  | 2.17 | 1.15     | 0.70 | 1.35  | 2.72  | 1.03   | 1.49 | 15m  | 623K   |
| GOF <sup>+</sup> [58]  | 0.81 | 1.08  | 0.91 | 0.83     | 0.78 | 0.89  | 0.52  | 0.70   | 0.82 | 1h   | 7.32M  |
| Nvdiffrac [33]         | 0.52 | 0.45  | 2.48 | 0.73     | 0.71 | 1.33  | 0.61  | 0.69   | 0.94 | >1h  | 81K    |
| Flexicubes [41]        | 0.46 | 0.44  | 5.23 | 0.55     | 0.70 | 1.05  | 0.26  | 0.68   | 1.17 | >1h  | 137K   |
| IMLS-Splat [55]        | 0.64 | 0.49  | 1.19 | 0.59     | 0.74 | 0.86  | 0.56  | 0.74   | 0.73 | 15m  | 578K   |
| Ours                   | 0.44 | 0.31  | 1.24 | 0.43     | 0.66 | 0.91  | 0.56  | 0.61   | 0.64 | 13m  | 216K   |

methods, including VolSDF [57], NeuS [51], and Neuralangelo [47]. (2) GS-based methods, including SuGaR [46], 2DGS [18], Triangle Splatting [16], GOF [58], PGSR [6], and QGS [61]. (3) Mesh-driven methods, including Nvdiffrac [33], FlexiCubes [41], IMLS-Splatting [55], and GeoSVR [26]. Our Nvdiffrac and FlexiCubes baselines utilize the Nvdiffrac framework with DMTet and FlexiCubes backends respectively. Note that our some methods [16, 18, 26, 33, 41, 47, 51, 55, 61] require or benefit from object masks on DTU. For fairness, masks are provided to those methods that benefit from them.

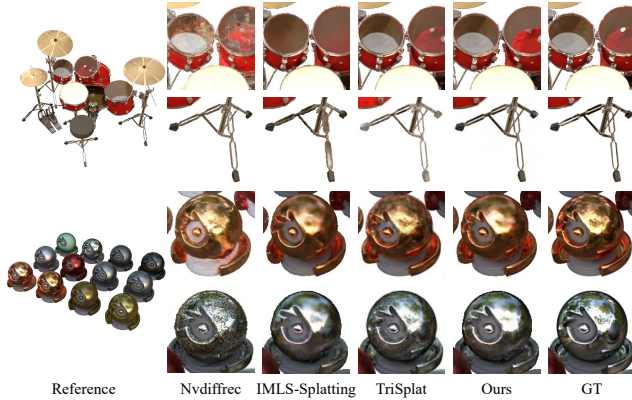


Figure 7. Qualitative rendering comparison on the NeRF-synthetic dataset. Compared to other mesh-driven methods, our method produces higher-fidelity results with more detailed textures.

**Geometry Evaluation.** To evaluate the geometric quality of the reconstructed meshes, we use Chamfer Distance (CD) for quantitative comparison. Results on DTU dataset are shown in Table 1. For our method, the reported runtime includes both initialization and training. Our method achieves geometric accuracy comparable to SOTA methods while demonstrating a significant advantage in efficiency, requiring fewer faces and shorter training time, which can also be observed in Figure 5. GeoSVR [26] and PGSR [6] generate meshes with a large number of redundant faces and visible floating artifacts. While meshes from QGS [61] and IMLS-Splatting [55] also show rough edges and suspended mesh fragments. In contrast, ExMesh produces meshes without obvious artifacts, recovering details while maintaining smooth and intact surfaces.

Compared to DTU dataset, the NeRF-synthetic dataset is more challenging due to its complex topology and diverse reflective properties. As shown in Table 2, our method

Table 3. Rendering quality results on the NeRF-synthetic dataset.

|             | Method          | PSNR  | SSIM  | LPIPS |
|-------------|-----------------|-------|-------|-------|
| GS-based    | 3DGS [23]       | 33.32 | 0.969 | 0.030 |
|             | 2DGS [18]       | 33.07 | 0.968 | 0.031 |
|             | TriSplat [16]   | 33.87 | 0.97  | 0.03  |
| Mesh-driven | Nvdiffrac [33]  | 26.87 | 0.930 | 0.090 |
|             | Flexicubes [41] | 27.50 | 0.930 | 0.080 |
|             | IMLS-Splat [55] | 28.38 | 0.950 | 0.060 |
|             | Ours            | 29.32 | 0.958 | 0.051 |

achieves geometric accuracy comparable to state-of-the-art methods. Figure 6 shows the visual comparison on the NeRF-synthetic dataset. 2DGS [18] and GOF [58] tend to produce multi-layered suspended mesh artifacts and introduce significant noise when fitting slender structures. Meshes reconstructed by Nvdiffrac [33] exhibit obvious polygonal faceting. In comparison, our method reconstructs smoother surfaces and captures fine engraved details that are not present in the results of other mesh-driven methods.

**Rendering Evaluation.** We further evaluate the novel view synthesis quality on the NeRF-synthetic dataset. As shown in Table 3, our method outperforms other mesh-driven methods on all rendering metrics (PSNR, SSIM [52], LPIPS [59]). The qualitative comparison in Figure 7 also shows that, compared to Nvdiffrac [33] and IMLS-Splatting [55], our rendered results exhibit higher fidelity and more detailed textures. However, there is still a gap in rendering quality when compared to Gaussian-based methods, as they leverage Spherical Harmonics (SH) to flexibly capture complex view-dependent appearance. Bridging this rendering gap for explicit meshes remains an ongoing challenge.

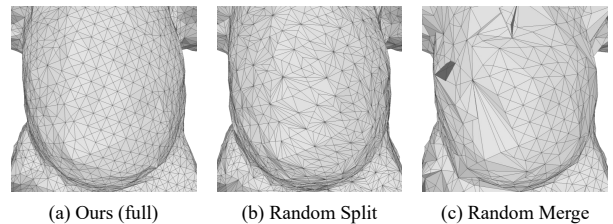


Figure 8. Ablation study on our vertex split and merge strategies.

### 4.3. Ablation Study

**Vertex Splitting and Merging.** We first validate effectiveness of the dynamic topology adjustment by comparing five variants: (1) only merge; (2) only split; (3) random split, where the edge to split is selected by our criterion but the split position is set to a random point; (4) random merge; and (5) our full method. Quantitative results on the NeRF-synthetic dataset are shown in Table 4. ‘‘Only Merge’’ fails to add faces for detailed fitting. The ‘‘Only Split’’ variant, while able to add details, led to a substantial increase in face count. Random strategies also perform worse than our full method. Qualitative comparison in Figure 8 intuitively reveals the reasons. Random split disrupts the local regularity

Table 4. Quantitative results for the ablation study on our vertex splitting and merging strategies on the NeRF-synthetic dataset.

| Setting      | CD↓  | PSNR↑ | Face Count |
|--------------|------|-------|------------|
| Only Split   | 0.74 | 28.77 | 239K       |
| Only Merge   | 1.81 | 23.51 | 12K        |
| Random Split | 0.77 | 28.30 | 184K       |
| Random Merge | 1.34 | 26.27 | 192K       |
| Ours (full)  | 0.64 | 29.32 | 196K       |

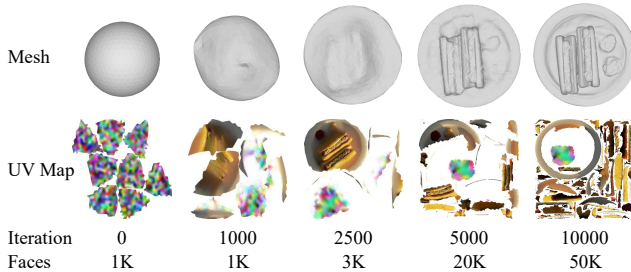


Figure 9. Ablation study on the initialization strategy.

of the mesh, leading to an uneven and chaotic triangulation. Additionally, random merge incorrectly collapses edges, introducing severe topological irregularities.

**Initialization Strategy.** Instead of using the initial coarse mesh described in Section 4.1, we use a spherical mesh with 1000 faces and random colors as the initial input. As shown in Figure 9, our full optimization pipeline can still deform and subdivide the mesh, eventually reconstructing the target object. We observe that, compared to our standard initialization strategy, training from an unrelated sphere requires more iterations to converge, and the final geometric accuracy is slightly lower. These results demonstrate that our method does not strictly depend on the initial mesh.

Table 5. Ablation study on texture representation.

| Setting          | CD↓  | PSNR↑ | Face Count |
|------------------|------|-------|------------|
| Per-Vertex Color | 0.76 | 28.47 | 356K       |
| Per-Face Color   | 0.78 | 28.12 | 442K       |
| MLP Texture      | 0.64 | 28.87 | 198K       |
| Ours (UV Map)    | 0.64 | 29.32 | 196K       |

**UV Mapping.** Moreover, our standard method is evaluated alongside three alternatives: (1) Per-Vertex Color, which stores color directly on the vertices; (2) Per-Face Color; and (3) Coordinate-based MLP textures. As shown in Table 5, binding texture to geometry (per-vertex/face) requires a significantly denser mesh to capture high-frequency details and match our geometric accuracy. In contrast, our decoupled strategy achieves similar fidelity with far fewer faces. Furthermore, compared to coordinate-based MLP textures, our UV-based method yields comparable quality while maintaining an editable texture map throughout optimization, avoiding post-training baking.

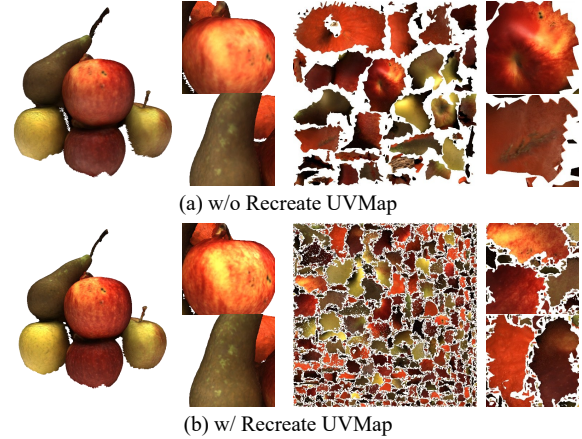


Figure 10. Ablation study on UV reconstruction.

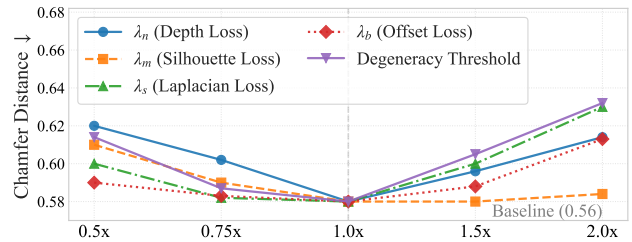


Figure 11. Ablation on hyperparameter sensitivity. The geometric accuracy remains highly stable as each parameter varies within  $[0.5\times, 2.0\times]$  of its default value.

**UV Reconstruction.** We further evaluate the necessity of periodic UV map reconstruction by comparing our standard strategy with creating UV map only once at the beginning. As shown in Figure 10, initial packing efficiency of UV islands is low. As the topology changes during training, a fixed UV map becomes inconsistent with the mesh, leading to texture stretching and UV island fragmentation. This forces fine details into small or distorted UV regions, limiting texture resolution and causing blurry results.

**Hyperparameter Robustness.** Finally, we conduct a comprehensive ablation study on five key hyperparameters, including the degeneracy threshold and various loss weights. As shown in Figure 11, the Chamfer Distance on the DTU dataset remains highly stable as each parameter varies within  $[0.5\times, 2.0\times]$  of its default value.

## 5. Conclusion

In this paper, we propose ExMesh, an explicit mesh reconstruction framework designed to address the critical challenge of balancing high fidelity and conciseness. By integrating differentiable optimization with discrete topology updates and real-time UV maintenance, our method achieves coarse-to-fine mesh refinement. While our current framework demonstrates robustness across hyperparameter settings, exploring self-adaptive optimization mechanisms remains a promising direction for future work.

## 6. Acknowledgments

This work was supported by the Open Fund of National Key Laboratory of Deep Space Exploration (Grant NKD-SEL2025008), the National Natural Science Foundation of China (Grant No. 62306294), and the Key Technology Research Project of TW-3 (TW3004).

## References

- [1] Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michal Iwanicki, and Adam Hargreaves. *Real-Time Rendering*. AK Peters/CRC Press, 4 edition, 2018. 1
- [2] Ma Baorui, Han Zhizhong, Liu Yu-Shen, and Zwicker Matthias. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. In *ICML*, 2021. 2
- [3] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *ACM SIGGRAPH*, 1998. 1
- [4] Edwin Catmull. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978. 1
- [5] Hanzhi Chang, Ruijie Zhu, Wenjie Chang, Mulin Yu, Yanzhe Liang, Jiahao Lu, Zhuoyuan Li, and Tianzhu Zhang. Mesh-splat: Generalizable sparse-view surface reconstruction via gaussian splatting. In *AAAI*, 2026. 3
- [6] Dong Chen, Hao Li, Wen Ye, et al. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE TVCG*, 31(9):6100–6111, 2024. 2, 3, 6, 7, 4
- [7] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *NeurIPS*, 2019. 3
- [8] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM SIGGRAPH*, 1996. 2, 3, 6
- [9] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH*, 2024. 3
- [10] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *ACM SIGGRAPH*, 1999. 6, 2
- [11] Y Dou, Z Zheng, Q Jin, et al. Differentiable micro-mesh construction. In *CVPR*, 2024. 2
- [12] Qianqian Fu, Qingsong Xu, Yew Soon Ong, Zexiang Wang, Lingjie Liu, and Christian Theobalt. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. In *NeurIPS*, 2022. 1, 2
- [13] Lin Gao, Jie Yang, Tong Wu, Yu-Kun Lai, Paul L. Rosin, and Ligang Liu. Tmnet: Deep texture mapping for 3d reconstruction. In *ICCV*, 2019. 2, 3
- [14] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH*, 1997. 4
- [15] J Held, S Son, R Vandeghen, et al. Meshsplatting: Differentiable rendering with opaque meshes. *arXiv preprint arXiv:2512.06818*, 2025. 2
- [16] J Held, R Vandeghen, A Deliege, et al. Triangle splatting for real-time radiance field rendering. In *3DV*, 2025. 2, 3, 6, 7, 4
- [17] Hugues Hoppe. Progressive meshes. In *ACM SIGGRAPH*, 1996. 4
- [18] Bo Huang, Zexiang Yu, Anpei Chen, et al. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH*, 2024. 2, 3, 6, 7, 4
- [19] JF Hughes, Andries van Dam, James D Foley, Steven K Feiner, and Kurt Akeley. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2 edition, 1990. 1
- [20] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanaes. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 6, 4
- [21] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018. 3, 6
- [22] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM TOG*, 32(3):29:1–29:13, 2013. 3
- [23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In *ACM SIGGRAPH*, 2023. 2, 3, 6, 7
- [24] Samuli Laine, Tero Karras, Miika Aittala, and Jaakko Lehtinen. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 39(6), 2020. 3, 6
- [25] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérôme Mailliot. Least squares conformal maps for automatic texture atlas generation. In *ACM SIGGRAPH*, 2002. 6
- [26] J Li, J Zhang, Y Zhang, et al. Geosvr: Taming sparse voxels for geometrically accurate surface reconstruction. In *NeurIPS*, 2026. 2, 3, 6, 7, 4
- [27] H Lin, S Chen, J Liew, et al. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025. 6, 2
- [28] Shichen Liu, Weikai Li, Tianye Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019. 3, 6
- [29] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH*, 1987. 1, 2, 3
- [30] X Lyu, Y T Sun, Y H Huang, et al. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *ACM TOG*, 43(6):1–12, 2024. 3
- [31] 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. 2, 6
- [32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4), 2022. 2
- [33] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Kayvon Fatahalian, Miika Aittala, and Samuli Laine. Extracting triangular 3d models, materials, and lighting from images. In *CVPR*, 2022. 2, 3, 5, 6, 7, 4

- [34] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 3
- [35] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020. 2
- [36] Michael Oechsle, Songyou Peng, Michael Niemeyer, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021. 2
- [37] Wolfgang Palfinger. Continuous remeshing for inverse rendering. *Computer Animation and Virtual Worlds*, 33(5): e2101, 2022. 3
- [38] Jiaxiang Shang, Yiyi Liao, Simon Green, Andreas Geiger, and Michael Niemeyer. Meshsdf: Differentiable iso-surface extraction. In *NeurIPS*, 2020. 2
- [39] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: Fast and robust angle based flattening. *ACM TOG*, 24(2):311–330, 2005. 6
- [40] Tianchang Shen, Jun Gao, Yichao Zhou, Zexiang Wu, Jacob Munkberg, Jon Hasselgren, Kayvon Fatahalian, Miika Aittala, Samuli Laine, Sergey Tulyakov, et al. Dmtet: Deep marching tetrahedra for 3d reconstruction. In *CVPR*, 2021. 2
- [41] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Jun Gao, Wenzheng Chen, Alex Evans, Kayvon Fatahalian, Miika Aittala, and Samuli Laine. Flexible isosurface extraction for gradient-based mesh optimization. *ACM TOG*, 42(4):1–16, 2023. 2, 3, 6, 7
- [42] Seung Wook Son, Matheus Gadelha, Yichao Zhou, Jacob Munkberg, Jon Hasselgren, Kayvon Fatahalian, Miika Aittala, and Samuli Laine. Dmesh: A differentiable mesh representation. *arXiv preprint arXiv:2404.13445*, 2024. 3
- [43] Seung Wook Son, Matheus Gadelha, Yichao Zhou, Jacob Munkberg, Jon Hasselgren, Kayvon Fatahalian, Miika Aittala, and Samuli Laine. Dmesh++: An efficient differentiable mesh for complex shapes. In *ICCV*, 2025. 3
- [44] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *SGP*, 2007. 1
- [45] AG Stuart, L Morton, I Stavness, et al. 3dgs-to-pc: 3d gaussian splatting to dense point clouds. In *ICCV*, 2025. 3
- [46] Shuaifeng Su, Peng Wang, Lingjie Liu, and Christian Theobalt. Sugar: Surface-aligned gaussian splatting for 3d reconstruction. In *ICCV*, 2023. 2, 3, 6, 7, 4
- [47] Matthew Tancik, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 1, 2, 6, 7, 3, 4
- [48] Gabriel Taubin. A signal processing approach to fair surface design. In *ACM SIGGRAPH*, 1995. 6, 2
- [49] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *ACM SIGGRAPH*, 1987. 1
- [50] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 2, 3
- [51] Peng Wang, Lingjie Liu, Yuan Lin, Xin Tong, Ziwei Liu, and Christian Theobalt. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 1, 2, 6, 7, 3, 4
- [52] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 7, 2
- [53] Y Wolf, A Bracha, and R Kimmel. Gs2mesh: Surface reconstruction from gaussian splatting via novel stereo views. In *ECCV*, 2024. 3
- [54] B Yang, C Bao, J Zeng, et al. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *ECCV*, 2022. 2
- [55] K Yang, L Dai, I Liu, et al. Imls-splatting: Efficient mesh reconstruction from multi-view images via point representation. *ACM TOG*, 44(4):1–11, 2025. 2, 3, 6, 7, 4
- [56] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. 2
- [57] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 1, 2, 6, 7, 3, 4
- [58] Zexiang Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM TOG*, 43(6):1–13, 2024. 2, 3, 6, 7, 4
- [59] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7
- [60] Xinyu Zhang, Yasir Siddiqui, Armen Avetisyan, et al. Vertexregen: Mesh generation with continuous level of detail. In *ICCV*, 2025. 3
- [61] Zhen Zhang, Bo Huang, Hao Jiang, et al. Quadratic gaussian splatting: High quality surface reconstruction with second-order geometric primitives. In *ICCV*, 2025. 2, 3, 6, 7, 4