Efficient and Structure-Aware 3D Reconstruction via Differentiable Primitive Abstraction

Gaoyang Zhang blurgy@zju.edu.cn
Yingxi Chen 22251126@zju.edu.cn
Hanchao Li hanson_li@zju.edu.cn
Xinguo Liu xinguoliu@zju.edu.cn

State Key Lab of CAD&CG, Zhejiang University
Hangzhou, China

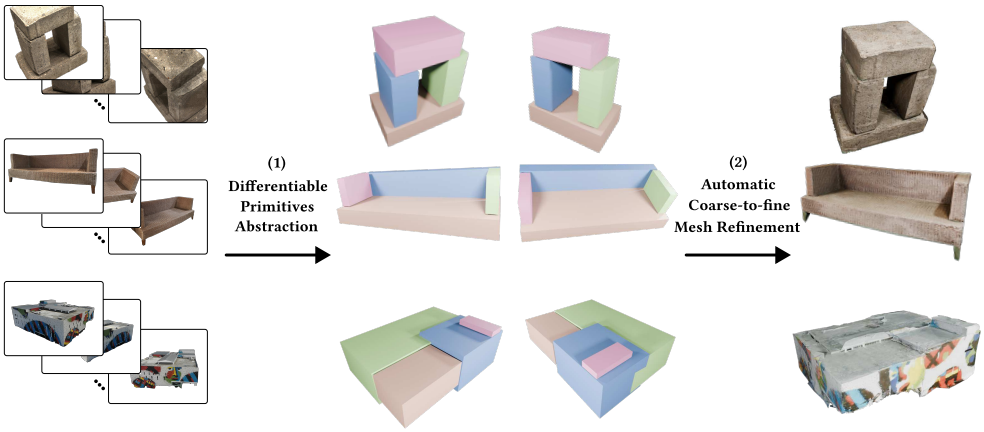


Figure 1: **Left**: Input multi-view images. **Middle**: Part-segmented mesh via differentiable primitive abstraction (different colors denote distinct parts). **Right**: Final high-fidelity textured mesh after automatic coarse-to-fine refinement.

Abstract

Reconstructing detailed 3D models from multi-view images often involves a trade-off between efficiency and fidelity. Existing methods based on volumetric representations or dense meshes can be computationally expensive, while primitive-based methods struggle to capture fine geometric details. We propose a novel method that addresses this challenge by combining differentiable primitive abstraction with adaptive mesh refinement. Our method first abstracts the scene into a set of cuboid primitives represented by analytical signed distance functions (SDFs), enabling part separability. This stage leverages differentiable volume rendering to efficiently optimize the primitives' poses and sizes. Subsequently, an automatic coarse-to-fine refinement process, guided by rendering error, restores fine geometric details. Our approach yields high-quality, part-separable meshes with manageable complexities, suitable for applications requiring part manipulation and efficient rendering. We demonstrate the effectiveness of our method on the DTU, BlendedMVS, and Tanks&Temples datasets, achieving a better balance between mesh complexity and reconstruction fidelity compared to existing techniques.

***Keywords:** 3D Reconstruction, Differentiable Rendering, Mesh Refinement, Neural Surface Reconstruction, Primitive Abstraction*

1. Introduction

Structure-aware reconstruction of 3D scenes has been one of the most fundamental problems in computer vision and graphics [26, 36]. In recent years, neural rendering [25, 34] has fueled a surge of works that enable high-fidelity 3D reconstructions [7, 11, 17, 44, 45, 49]. While these methods have achieved impressive results in capturing intricate details and complex geometries, they often rely on computationally expensive volumetric representations or dense meshes. This work tackles the problem of efficiently reconstructing detailed 3D models from multi-view images that are both lightweight and inherently part-separable, addressing the limitations of existing methods regarding computational cost, part awareness, and practical applicability

in real-time or resource-constrained scenarios, such as interactive AR/VR applications and robotics.

The core challenge lies in balancing the efficiency of representation with the fidelity of the reconstruction. Primitive-based methods offer efficiency and part separability, but often struggle to capture fine geometric details due to the limited expressiveness of simple primitives [27, 46]. Conversely, methods based on neural implicit representations, such as those utilizing signed distance functions (SDFs) [44, 49] or occupancy fields [24], excel at representing complex shapes, but often suffer from high computational costs due to their reliance on volumetric rendering or computationally expensive neural networks. Moreover, these methods typically lack explicit part segmentation, hindering their applicability in tasks requiring part-based manipulation or analysis. Our method addresses these challenges by combining the strengths of primitive-based and neural implicit representations. Unlike existing methods that rely on either computationally intensive volumetric representations or explicit primitive meshes, our approach leverages a novel combination of analytical implicit primitives representation and an adaptive mesh refinement scheme. This allows us to benefit from the efficiency and part-awareness of primitives during the initial abstraction stage while achieving high-fidelity through a targeted refinement process that focuses on areas of high geometric detail.

We propose a two-stage pipeline that first enables part separability by abstracting the scene into a set of cuboid primitives represented by analytical SDFs, and subsequently refines the resulting mesh explicitly using an adaptive, coarse-to-fine strategy. As shown in Fig. 1, this approach enables efficient capture of the overall structure while providing the flexibility to refine details where necessary.

We demonstrate the effectiveness of our method on scenes from DTU [12], BlendedMVS [48], and Tanks&Temples [15] datasets, achieving a better balance between mesh size and reconstruction fidelity than previous methods. Our robust primitives abstraction also extrapolates more plausible novel views in unobserved areas, and enables part-based manipulation of the final reconstruction. The main technical contributions of this work are:

- A two-stage reconstruction pipeline that achieves both part separability and preservation of geometric details.
- A differentiable formulation for fitting analytical SDF-based primitives to a scene represented by multi-view images, enabling efficient optimization of part-based representations.
- An automatic coarse-to-fine mesh refinement strategy that optimally allocates detail, resulting in lightweight yet accurate meshes.
- We demonstrate improved reconstruction accuracy as indicated by Chamfer distance with more manageable mesh complexities, and ability to do view extrapolation and part-based manipulation.

2. Related Work

Our work draws inspiration from research in neural rendering, implicit scene representations with SDF, and primitive-based 3D reconstruction.

Neural Rendering. Differentiable rendering techniques enable the optimization of scene parameters by back-propagating gradients through the rendering process. Early work focused on approximating the gradients of discrete rasterization [14] or using probabilistic contributions of mesh facets [18]. Later methods explored differentiable volumetric rendering with exact derivatives for surface intersection [29]. These advances in differentiable rendering have paved the way for optimizing mesh geometry and other scene parameters directly from images. Our work leverages differentiable rendering for efficiently optimizing the poses and sizes of primitives during the abstraction stage and for refining vertex positions during the mesh refinement stage.

Neural Implicit Representations for 3D Reconstruction. Neural rendering techniques, particularly NeRF [25], have significantly advanced novel view synthesis. However, NeRF-based methods [3–5] often lack explicit definition of surface geometry, posing challenges for mesh extraction. Techniques like marching cubes [21] can extract meshes from implicit representations like NeRF, but frequently result in dense meshes with high polygon counts and sub-optimal detail preservation. NeuS [44] and its concurrent work [30, 49] proposes to directly learn a signed distance function (SDF) or occupancy representation using differentiable rendering, achieving high-fidelity reconstructions. However, they typically rely on computationally expensive coordinate-based MLPs to predict the field value, and does not address part separability. Our method adopts the SDF-based representation, but instead of predicting the SDF with an MLP, we use a set of analytical primitive SDFs to obtain the signed distance at arbitrary spatial position, and optimize the poses of the primitives. This approach allows us to retain the expressiveness of the SDF representation with drastically reduced computational cost.

Primitive-Based 3D Reconstruction. Reconstructing scenes with simple geometric primitives like points, segments, planes, or cuboids and beyond has been studied for a long time. Points and line segments are often fitted via multi-view stereo [19, 38, 39]. Plane fitting can be achieved by estimating planar parameters from depth captures [9],

sparse views [13], or monocular RGB sequences [47]. Another line of works extract solid shapes from inputs using learning-based methods [40, 43]. Methods based on primitive shapes offer efficiency and part separability, but struggle to capture fine geometric details [20, 27, 33, 46, 51]. To mitigate this, we opt to marry traditional mesh refinement techniques with an automatic coarse-to-fine refinement strategy enabled by recent neural rendering methods, adaptively adding details to a primitive-based initial mesh according to rendering error.

3. Method

Given a set of multi-view RGB images of a target object, our goal is to obtain a triangular mesh that is *lightweight*, *part-separable* and enables *high-fidelity* rendering. To achieve this, we leverage analytical SDFs of cuboid primitives to support native part-based mesh export, and devise an automatic coarse-to-fine mesh refinement stage to restore the delicate geometric details and texture.

We begin with the necessary background knowledge on differentiable rendering and implicit surface representations that our work builds upon (Sec. 3.1). We then elaborate on the two essential stages of our pipeline: (1) differentiable primitive abstraction (Sec. 3.2) and (2) adaptive mesh refinement (Sec. 3.3). These two stages work in tandem to produce lightweight, part-separable, and high-fidelity 3D meshes from multi-view images.

Notations. Given a sequence of images $\mathcal{I}$, we denote the sparse point cloud estimated via SfM as $\mathcal{P}$. Our method first optimize m cuboid primitives $\{G_1, \cdots, G_m\}$ parameterized by their respective bounds and pose matrices:

$$G_i = g(\mathbf{B}_i, [\mathbf{R}_i, \mathbf{T}_i]), \tag{1}$$

where $\mathbf{B}_i \in \mathbb{R}^3$ parameterizes the side lengths of the cuboids in their local coordinate system, and $[\mathbf{R}_i, \mathbf{T}_i] \in \mathbb{R}^{4\times3}$ parameterizes the rigid transformation between the world coordinate system and their own.

3.1. Preliminary

Our pipeline is enabled by several recent advancements in differentiable rendering.

Neural Radiance Fields. NeRF [25] models scenes using volume density and view-dependent emitted radiance at each spatial location. The NeRF function takes a 5-D input $(\mathbf{x}, \mathbf{d}) = [(x, y, z), (\theta, \phi)]$ and outputs a volume density δ and color (r, g, b), which are then used in differentiable volume rendering:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))f_{\mathbf{c}}(\mathbf{r}(t), \mathbf{d})dt, \tag{2}$$

where $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ is a ray sampled from the input image collection $\mathcal{I}$, $T(t) = \exp(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds)$ is the transmittance value at ray position $\mathbf{r}(t)$, $f_{\mathbf{c}}$ is a function that estimates the emitted radiance at $\mathbf{r}(t)$.

Parameterizing SDFs for Volume Rendering. While NeRFs use volume density for radiance integration along camera rays, such density is not well-defined for mesh extraction using marching cubes [21]. [50] uses a surface rendering technique that only considers a single surface intersection point for each ray, which can lead to poor local minima and struggles to reconstruct complex geometry with self-occlusions or thin structures. Later works explored representing scene geometry with neural fields (signed distance functions [44, 49] and occupancy functions [30]) to exploit the expressivity of implicit functions for modeling scene geometry. We adopt the parameterization from Wang *et al*. [44] in our method, but theoretically our method should also be compatible with other SDF-to-density parameterizations.

3.2. Differentiable Primitive Abstraction

Given a collection of RGB images of a central object from different viewing directions, the goal of this stage is to extract a set of cuboid primitives that captures the coarse overall geometry of the underlying object.

Scene Parameterization. We use a composition of cuboid primitives to represent the scene geometry. Each cuboid primitive G_i is represented as an *analytical* signed distance function parameterized by a bounds vector $\mathbf{B}_i \in \mathbb{R}^3$ and a rigid transformation $[\mathbf{R}_i; \mathbf{T}_i] \in \mathbb{R}^{4\times3}$. With this representation, the SDF of any point $\mathbf{x} \in \mathbb{R}^3$ from the scene can be trivially computed as the $\min$ of the SDF value evaluated separately at each primitive:

$$\text{SDF}(\mathbf{x}) = \min_{i=1}^{m} \text{SDF}_{G_i}(\mathbf{x}; \mathbf{B}_i, \mathbf{R}_i, \mathbf{T}_i), \tag{3}$$

where $\text{SDF}_{G_i} \in \mathbb{R}$ is parameterized by the bounds $\mathbf{B}_i$ and pose $[\mathbf{R}_i, \mathbf{T}_i]$ of the i-th primitive G_i. This parameterization is differentiable w.r.t. the primitive bounds $\mathbf{B}_i$ and pose $[\mathbf{R}_i, \mathbf{T}_i]$. The scene appearance is modeled separately using a shallow MLP with a spatial hash grid [28]. The radiance estimator $f_{\mathbf{c}}$ from Eq. (2) in our case can be denoted as

$$f_{\mathbf{c}}(\mathbf{r}(\mathbf{t}), \mathbf{d}, \mathbf{n}, \text{SDF}(\mathbf{r}(t))), \tag{4}$$

where $\mathbf{n}$ is the predicted normal direction at position $\mathbf{x}$, and can be computed trivially as $\mathbf{n} = \nabla_{\mathbf{x}}\text{SDF}(\mathbf{x})$ [10].

The formulation based on analytical primitive SDFs is inspired by [44], which uses a neural network to predict a point $\mathbf{x}$'s SDF value. We instead use a combination of

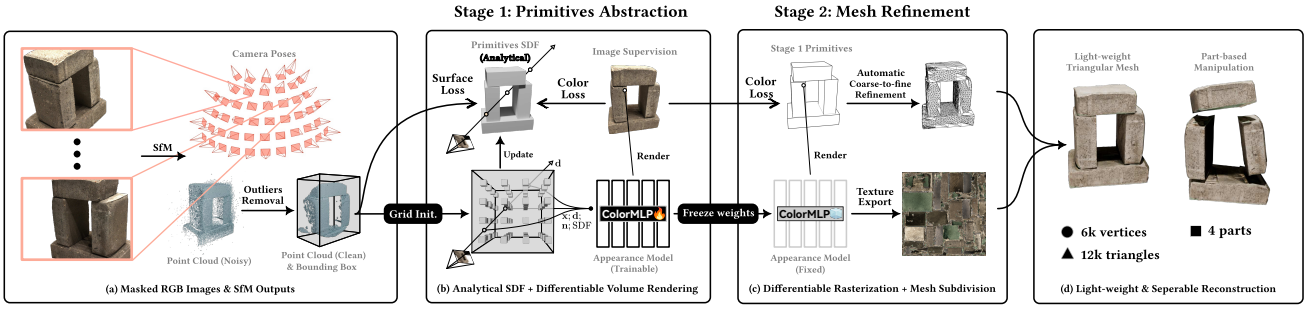


Figure 2: **Overview of the proposed pipeline. (a)** The input to the pipeline is a collection of RGB images. Through structure-from-motion, we obtain camera poses and a sparse point cloud. The point cloud is further cleaned up to remove the outliers for use in the next step in the pipeline. **(b)** The configuration of the set of cuboids is initialized and jointly optimized with an appearance model, resulting in an initial scene representation. **(c)** The automatic coarse-to-fine mesh refinement procedure restores delicate details overlooked by the primitives-based representation. **(d)** The final output is a light-weight, part-separable mesh with textures enabling high-fidelity rendering.

analytical primitive SDFs and optimize their poses, this approach is much more parameter-efficient and thus faster to evaluate and optimize. The primitives configuration can also be exported directly into a triangular mesh with minimal polygon facets (*e.g.* top left of Fig. 2 (c)), avoiding an unnecessarily excessive amount of triangles caused by marching cubes [21], which has been adopted by previous works [17, 25, 49] for extracting geometry.

Primitives Initialization. With the scene parameterization, we first need an initialization to facilitate efficient optimization. As shown in Fig. 2 (a), the input to our pipeline is a collection of RGB images. We use an off-the-shelf structure-from-motion toolkit [38] to obtain the camera parameters and an initial sparse point cloud $\mathcal{P}$. Subsequently, the outliers in $\mathcal{P}$ are filtered out in two steps: (1) the points that deviates significantly from the surface points are removed using local point cloud characteristics [37]. (2) the points that are visible by no more than 50% of the input views are removed. This results in a refined, outlier-free point cloud $\mathcal{P}_{\text{in}}$. We initialize a set of primitives within a regular grid filling the bounding box of $\mathcal{P}_{\text{in}}$. Specifically, 64 (4^3) cuboid primitives are uniformly distributed within this bounding box, as visualized in Figure 2 (b). The subsequent optimization process effectively prunes redundant primitives, as detailed next.

Scene Optimization & Regularization. For a given ray $\mathbf{r}$ sampled from the input images, we render a pixel's color value using Eq. (2). The scene representation is supervised directly with an image reconstruction loss:

$$L_{\text{render}} = \mathbb{E}_{\mathbf{r}\sim\mathcal{I};t}\|C(\mathbf{r}) - \hat{C}(\mathbf{r})\|_2. \tag{5}$$

We also introduce a surface loss term to encourage alignment of the primitives with the cleaned point cloud $\mathcal{P}_{\text{in}}$:

$$L_{\text{surface}} = \mathbb{E}_{\mathbf{p}\sim\mathcal{P}_{\text{in}}}|\text{SDF}(\mathbf{p})|. \tag{6}$$

To prevent the optimization process from placing multiple primitives at the same location, for each pair of primitives that are overlapping in space are encouraged to shrink with a novel pruning loss. Formally, we compute between each pair of primitives the overlapping ratio $o(i, j)$, and define the pruning loss for the i-th primitive as:

$$L_{\text{prune}}(i) = \begin{cases} \sum \mathbf{B}_i, & \exists\, j \text{ s.t. } o(i,j) > \tau, \prod \mathbf{B}_i < \prod \mathbf{B}_j \\ 0, & \text{otherwise} \end{cases}, \tag{7}$$

with τ being a predefined overlapping threshold. This loss shrinks a smaller, enclosed primitive until it disappears, effectively pruning it. The overall pruning loss for the scene is summed over all of the m primitives:

$$L_{\text{prune}} = \mathbb{E}_{i\sim\mathcal{U}(\{1,\cdots,m\})} L_{\text{prune}}(i). \tag{8}$$

Our scene representation is supervised by minimizing the combined loss:

$$L = L_{\text{render}} + \lambda_1 L_{\text{surface}} + \lambda_2 L_{\text{prune}}. \tag{9}$$

As shown in Fig. 2 (b), we can recover the overall geometry represented as a set of primitives with optimal poses (top left) and a volumetric scene appearance model (bottom right) after this optimization stage.

3.3. Automatic Coarse-to-Fine Mesh Refinement

While the primitives-based geometry has the nice property of being part-separable, it overlooks fine geometric details; on the other hand, the volumetric scene appearance model needs to be evaluated at multiple points of a

ray to compute the color of the corresponding pixel. To this end, we devise an automatic coarse-to-fine mesh refinement method that alternates between *vertices refinement* and *adaptive mesh detailization*, aiming at restoring the fine geometric details, finally exporting a texture map that enables high-quality rendering of the output mesh.

Vertices Refinement. Geometric detail lies in the precise arrangement of the vertices. We leverage differentiable rasterization [16] to render the mesh and refine vertex positions with a rendering loss similar to Eq. (5). During rasterization, the color of a pixel is determined by querying the frozen appearance model $f_{\mathbf{c}}$ with the rendered point $\mathbf{x}$ on the mesh and the negated triangle normal $-\mathbf{n}_{\text{tri}}$ as the viewing direction $\mathbf{d}$:

$$\mathbf{c}_{\text{pixel}} = f_{\mathbf{c}}(\mathbf{x}, -\mathbf{n}_{\text{tri}}, \mathbf{n}_{\text{tri}}, 0). \tag{10}$$

Vertex positions are then optimized to minimize the following rendering loss:

$$L = \mathbb{E}_{\text{pixel}\sim\mathcal{I}} \|\mathbf{c}_{\text{pixel}} - \hat{\mathbf{c}}_{\text{pixel}}\|_2 \tag{11}$$

Adaptive Mesh Detailization. The initial mesh is intentionally coarse to facilitate efficient refinement. We introduce a novel adaptive mesh detailization scheme that selectively increases mesh resolution in regions contributing most significantly to the rendering loss. For each camera-visible face, we accumulate the rendering error E_{face} throughout the optimization process. Every k optimization steps, we sort the E_{face} values in descending order and define a subdivision threshold $\text{e}_{\text{subdivide}}$ as the 90-th percentile:

$$\text{e}_{\text{subdivide}} = \text{percentile}(E_{\text{face}}, 90). \tag{12}$$

Faces with E_{face} exceeding this threshold undergo mid-point subdivision, generating a new mesh with refined vertex positions. This process iterates K times, progressively refining the mesh until the final output is obtained.

This coarse-to-fine strategy allocates increasing degrees of freedom to geometrically complex regions, as demonstrated in Figure 2 (c) (top right). Subdivision is concentrated in areas with fine details, such as corners and edges, while flatter regions retain a lower resolution consistent with their geometric complexity.

Comparison to Related Work. Our approach shares similarities with the work of Tang *et al*. [42], which refines a mesh using both subdivision and decimation. However, their method requires decimation because their initial mesh, extracted from a density-based representation [25,28], is excessively dense. In contrast, our method starts with a simplistic mesh, obviating the need for decimation. Another relevant work is Differentiable Blocks World (DBW) [27], which represents scenes using superquadric meshes. Despite the expressive power of superquadrics [2, 31], DBW faces a trade-off between optimization efficiency and geometric detail. Our method avoids this trade-off through adaptive refinement, efficiently allocating resources to complex regions while maintaining overall optimization efficiency. This results in a high-quality final mesh with detailed geometry and a texture map suitable for rendering.

Table 1: **Quantitative comparison of geometric reconstruction accuracy on the DTU dataset [12].** Chamfer distance (CD) between reconstructed mesh and ground-truth point cloud are reported. Lower is better. Best results, Second-best results.

Scene	NeRF2Mesh [42]	DBW [27]	Ours
S5	3.04	6.54	2.60
S10	2.51	1.69	1.75
S11	3.33	2.67	1.92
S12	1.83	2.93	1.64
S34	2.97	3.60	1.93
S40	3.06	1.23	1.82
S45	3.23	4.57	2.06
S126	3.28	1.73	1.87
Mean CD	2.90	3.12	1.95
#vertices↓	137.2k	0.3k	6.1k
#faces↓	255.9k	0.5k	12.3k

4. Experiments

4.1. Implementation Details

The input to our pipeline is a collection of multi-view RGB images. We use SAM2 [35] to segment the object of interest from the first image and propagate to other images, the whole annotation process takes around 30s for each scene. The camera poses and initial sparse point cloud are obtained from the annotated images using COLMAP [38]. In the differentiable primitive abstraction stage, we use a cosine warmup with restart learning rate schedule [22], with 3e-3 as the peak learning rate. We use $\lambda_1 = 0.1$ and $\lambda_2 = 0.07$ for the surface loss and pruning loss, respectively. The differentiable primitive abstraction stage is run for 6,144 steps with 6 learning rate restarts, taking about 3 minutes. The output from this stage is a simplistic mesh with $n \times 6 \times 2 = 12n$ triangular faces, where n is the number of remaining (not pruned) cuboid primitives. This mesh undergoes two iterations of mid-point subdivision, resulting in $12n \times 4^2 = 192n$ faces, before it is fed to the subsequent steps. For the automatic coarse-to-fine mesh refinement stage, we perform subdivision after every 256 optimization steps, and stop the refinement process after 10 minutes, re-

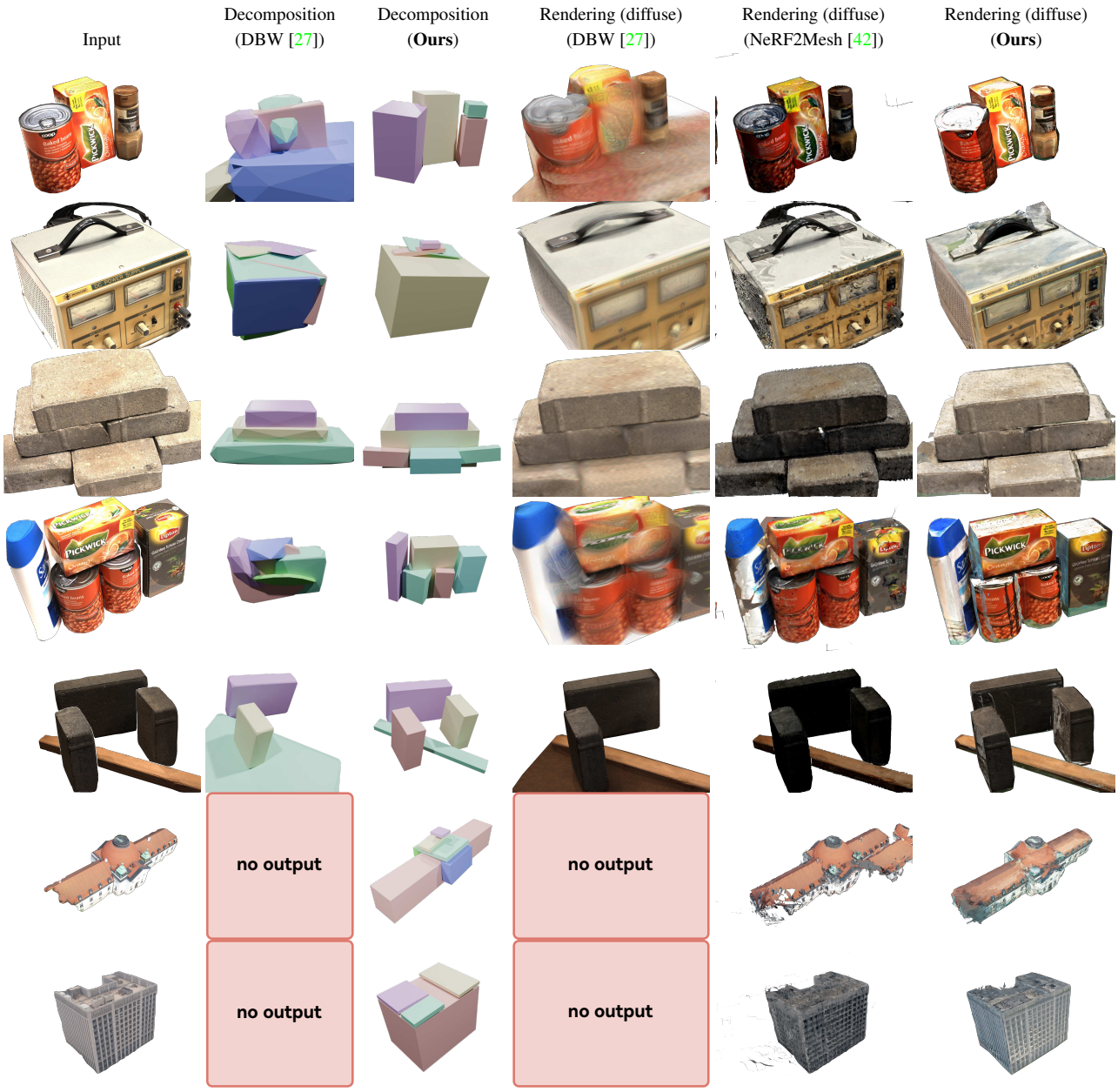


Figure 3: **Qualitative reconstruction results comparison on the DTU dataset [12].** Decomposed primitives are randomly colored for easier inspection. DBW [27] fails to produce meaningful outputs on in-the-wild data using their recommended parameters. NeRF2Mesh [42] reconstructions suffer from floaters and holes. Our method obtains clean and meaningful primitive decompositions as well as high-quality renderings. Please refer to the supplementary material for full reconstruction results.

sulting in ∼40 iterations of subdivision-optimization loops. In both stages, we use the AdamW [23] optimizer. The main experiments reported in this paper are run on a single RTX 4090 GPU, but since the peak GPU memory usage is only 8GiB, most commodity hardware can run our pipeline easily.

4.2. Results

Datasets and Metrics. We evaluate our method on the DTU MVS dataset [12]. The dataset contains 80 scenes captured with a 6-axis industrial robot in a controlled indoor setting, each scene consists of 49 or 64 accurate camera poses, and comes with 3D ground-truth points that are obtained with a structured light scanner. We evaluate on 8

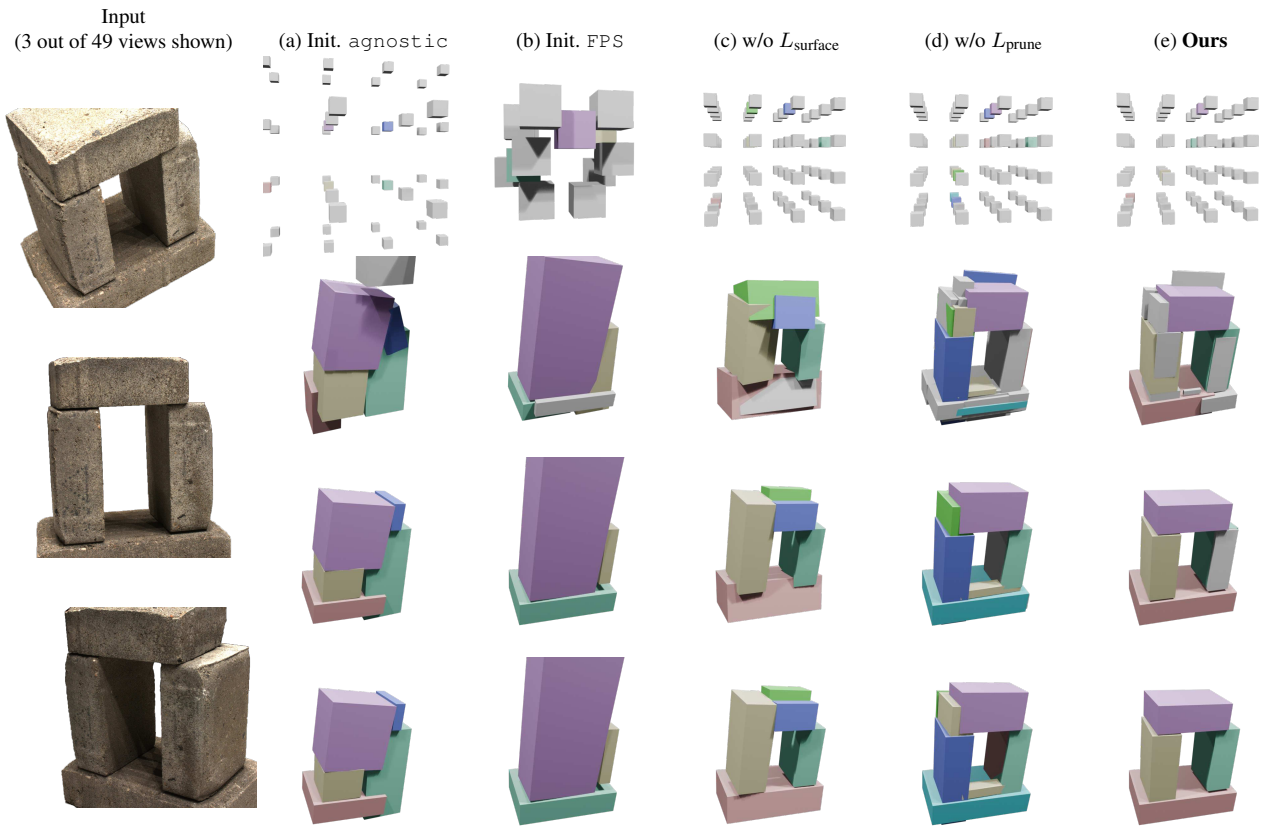


Figure 4: **Ablation study of the proposed differentiable primitive abstraction stage.** Four optimization timesteps $(0\%, 10\%, 50\%, 100\%)$ are visualized from top to bottom. Primitives are randomly colored for easier inspection. **(a)**, **(b)** Replacing our point cloud-based grid initialization with either a point cloud-agnostic one or a more sophisticated FPS-based initialization leads to the algorithm converging to a suboptimal local-minimum. **(c)** Removing the surface loss results primitives that are less faithful to the input scene. **(d)** Removing the pruning loss causes over-decomposition, where many primitives are placed in the same space. **(e)** Our full method converges faster and achieves better results. Please refer to Sec. 4.3 for detailed descriptions. The optimization progress of other scenes is provided in the supplementary video.

scenes that have different foreground objects and arrangements whose primitive decompositions are relatively intuitive. The official method by DTU is used for evaluating geometric reconstruction accuracy, where the Chamfer distance is computed between the ground-truth point cloud and points sampled from the reconstructed mesh. We also report the number of vertices and faces in the final output mesh as an indicator of mesh complexity.

Additionally, we select 3 scenes from the BlendedMVS [48] dataset and 1 scene from the Tanks&Temples [15] dataset, and a scene captured by a mobile phone as in-the-wild examples to test our method.

Baselines. We compare our method against two recent baselines closely related to our approach, namely NeRF2Mesh [42] and DBW [27]. NeRF2Mesh employs explicit mesh optimization similar to our method, but its input is from a density-based geometric representation, which can be noisy compared to our primitives based geometric representation. DBW directly optimizes a set of randomly initialized superquadric meshes to decompose a scene. We use the official implementations with recommended settings provided by the authors [27, 41] in our experiments.

Quantitative Results. Quantitative results on geometric reconstruction are reported in Tab. 1. As evident in Tab. 1, our method consistently outperforms the baselines in terms of Chamfer distance, indicating superior geometric reconstruction accuracy. On the one hand, although DBW's output mesh is the simplest according to number of vertices & faces, its 3D reconstruction accuracy is severely impacted by the over simplification, resulting in a large chamfer distance on more complex compared to NeRF2Mesh or our method. On the other hand, our method controls the

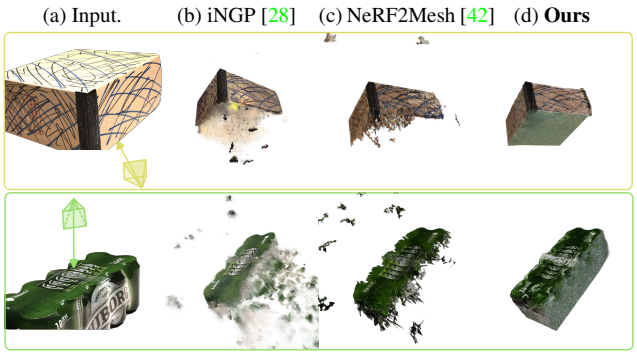


Figure 5: **View extrapolation.** When tested with views far from training views, our method maintains tight geometry with plausible extrapolated texture colors in unobserved areas.

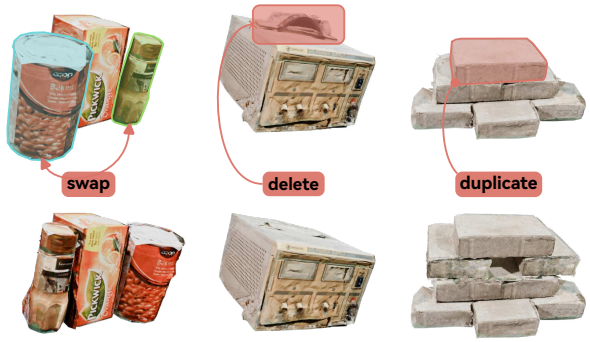


Figure 6: **Part-based manipulation of reconstructed meshes.** The reconstructed mesh is inherently part-based, and can be edited at ease using 3D modeling software like Blender [6].

mesh complexity at a manageable amount — it reconstructs meshes with 1-2 orders of magnitude fewer vertices & faces than NeRF2Mesh, in the meantime achieving the most accurate reconstruction, reaching a better balance between the efficiency of representation and the fidelity of the reconstruction.

Qualitative Results. We show the primitive abstractions and the final rendering results in Fig. 3. The primitive abstractions and the final textured reconstructions on all 13 scenes are provided in the supplementary material. The primitives decomposed by our method (second and third columns) are cleaner and more faithful than those of DBW across different scenes. Qualitative rendering results are shown in the last three columns of Fig. 3. DBW outputs overly blurred rendering due to its less faithful geometries, and fails on the more challenging in-the-wild examples. NeRF2Mesh's renderings suffer from holes and floaters due to its initially density-based geometric representation. In contrast, our renderings are much more satisfactory. This is thanks to both the high-quality primitive abstractions from the first stage, and the automatic coarse-to-fine mesh refinement strategy that focuses on adding intricate details to the mesh.

View Extrapolation. The primitives-based geometric representation serves as an implicit regularization, ensuring a clean reconstruction when the underlying scene is simple. In Fig. 5, we compare between Instant-NGP (iNGP) [28], NeRF2Mesh [42], and our method on view extrapolation performance. Instant-NGP, a NeRF variant focusing on fast and high-quality novel view synthesis, exhibits bogus geometry (floaters) [32] and implausible colors when tested with views far from training views. NeRF2Mesh also suffers from floaters and holes due to its initially poorly defined geometry derived from volume rendering density. In contrast, our method obtains a tight geometry with plausible extrapolated texture colors in areas unobserved during training.

Part-based Manipulation. Our differentiable primitive abstraction stage ensures that the resulting mesh is inherently part-separable. This enables a variety of downstream modeling tasks at the object-level, beyond traditional vertices/edges/faces-based mesh editing. In Fig. 6, we showcase three examples of part-based editing operations: swapping two objects in the scene (left), deleting the handle of the object by deleting the corresponding decomposed part (middle), and duplicating a block two times and placing the copies below the original (right). Part-based manipulation offers a more intuitive and efficient way to interact with and modify 3D models, opening up possibilities for applications like interactive design, virtual prototyping, and robotic manipulation.

4.3. Ablation Studies

In the following, we conduct controlled experiments on the DTU dataset [12] to validate various design choices of our pipeline.

Geometric Priors. Geometric priors are crucial for efficient optimization in the differentiable primitive abstraction stage. This includes efficient initialization from the outlier-free SfM point cloud, and using the surface loss to penalize the implicit field formed by primitives to stick to the geometry. We visualize the optimization dynamics of our primitive abstraction stage with various components removed in Fig. 4, and provide the optimization progress of all reported scenes in the supplementary video. A Blender [6] script used to generate the results for all scenes can be found in the supplementary material.

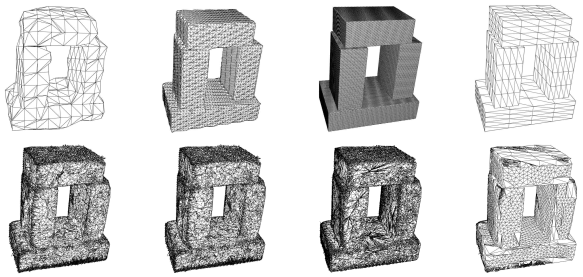
Table 2: **Ablation study of the losses in the first stage.** Removing either L_{surface} or L_{prune} leads to drop of reconstruction accuracy. Chamfer distance on the DTU dataset [12] is reported.

DTU Scene	S5	S10	S11	S12	S34	S40	S45	S126
w/o L_{surface}	2.74	2.12	2.54	1.69	2.08	2.59	2.49	2.40
w/o L_{prune}	2.92	2.10	1.93	1.79	**1.77**	2.33	2.32	1.99
Ours	**2.60**	**1.75**	**1.92**	**1.64**	1.93	**1.82**	**2.06**	**1.87**

We test the effectiveness of the point cloud based grid initialization by comparing with two alternative primitive initialization schemes: (1) `agnostic` initialization places primitives in a regular grid filling the $[-1, 1]^3$ bounding box predefined for all scenes, thus agnostic to each scene's point cloud; (2) `FPS` (farthest point sampling) [1, 8] initialization chooses the center of the initial primitives iteratively from the estimated point cloud, each time selecting a point that lies the farthest from all the chosen points, allowing for a uniform distribution of primitives that has a good coverage of the input point cloud. As shown in Fig. 4 (a) and (b), both alternative initialization methods fail to converge to a satisfying decomposition. The `agnostic` initialization places primitives too sparsely, causing the scene representation quickly falls into a local-minima that is hard to correct in later optimization steps. We find the `FPS` initialization to be unstable and sensitive to the shape of the point cloud. It places few primitives at locations with few points, however, the point cloud is estimated by SfM algorithms [38] and is likely missing parts that are observed by few cameras. We remove the surface loss L_{surface} and visualize the optimization process in Fig. 4 (c). The impact of removing the surface loss is evident, as its reconstruction accuracy is reported in Tab. 2. Without it, the set of primitives lacks explicit geometric constraints and can grow over the actual bounds to allow for better training-time rendering metrics, at the cost of a lower geometric reconstruction accuracy.

Pruning Loss. The pruning loss is crucial for our method to obtain a clean decomposition of the scene. It shrinks the smaller primitive when two primitives are placed in the same space. We disable this loss to better understand the impact it has on our primitive abstraction algorithm. As shown in Fig. 4 (d), the algorithm is prone to over-decomposition without L_{prune}, resulting in less meaningful decomposed parts. As reported in Tab. 2, the overall reconstruction accuracy is also reduced.

Coarse-to-Fine Mesh Refinement Strategy. The geometry representation based on analytical SDF of primitives from our first stage enables both marching cubes [21] based mesh extraction and analytically accurate mesh export. In our main experiments, we use the latter to generate a simplistic mesh as input for the subsequent mesh refinement stage. This allows us to focus on introducing the missing details overlooked by the first stage, and naturally provisions our automatic coarse-to-fine refinement strategy. Here we explore the use of marching cubes for bridging the two stages, generating a mesh with dense faces that is then refined using differentiable rasterization. Apart from surface subdivision, we also enable decimation&remeshing as done in [42] to allow the refinement stage to simplify the mesh. Qualitative result on `S40` from the DTU dataset is shown in Fig. 7. Using a mesh extracted with marching cubes, the initial face density is controlled by an extra resolution parameter `r` — a lower `r` leads to a poorly initialized mesh, limiting the performance of the refinement stage; a larger `r` increases the geometric accuracy of the mesh, but also increases its complexity. Instead, our method initializes the mesh from an analytical primitives representation, eliminating the tradeoff between simplicity and geometric accuracy.



MC, `r` = 32 #f = 157k | MC, `r` = 128 #f = 160k | MC, `r` = 512 #f = 200k | **Ours**, auto #f = **6k**

Figure 7: **Ablation study of the proposed automatic coarse-to-fine refinement strategy. Top row**: Input to the refinement stage, extracted by different methods and parameters. **Bottom row**: Refinement result. Increasing marching cubes (MC) resolution `r` could increase surface details, but also leads to an excessive number of faces (#f). Our method eliminates such tradeoff.

5. Limitations and Conclusion

Limitations. While our method demonstrates promising results with great robustness, there is still room for improvement. For instance, the current cuboid primitives may not adequately represent highly detailed objects with intricate curvatures. Future work could explore incorporating more sophisticated primitive shapes. Additionally, the texture generation process bakes the scene's lighting environment into the texture map. This could be improved by exploring more advanced techniques, such as material reconstruction or learned texture representations, to decouple the lighting information from the texture and enable more realistic and versatile rendering.

Conclusion. We have presented an efficient pipeline that reconstructs part-separable and high-quality textured meshes with a manageable face count from multi-view RGB images. Our pipeline leverages differentiable volume rendering and analytical primitive SDFs for efficient scene decomposition and an automatic coarse-to-fine strategy for detail refinement. The resulting meshes are suitable for a range of applications where part-based manipulation and efficient rendering are desired. We believe our method makes an important step towards efficient and practical, structure-aware 3D reconstruction from images, and we hope this work inspires further research in this direction.

6. Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 62032011).

References

[1] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, USA, Jan. 2007. Society for Industrial and Applied Mathematics. 9

[2] Barr. Superquadrics and Angle-Preserving Transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, Jan. 1981. Conference Name: IEEE Computer Graphics and Applications. 5

[3] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, Montreal, QC, Canada, Oct. 2021. IEEE. 2

[4] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields, Mar. 2022. arXiv:2111.12077 [cs]. 2

[5] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. TensoRF: Tensorial Radiance Fields, Nov. 2022. arXiv:2203.09517 [cs]. 2

[6] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 8

[7] P. Dai, J. Xu, W. Xie, X. Liu, H. Wang, and W. Xu. High-quality Surface Reconstruction using Gaussian Surfels, Apr. 2024. arXiv:2404.17774 [cs] version: 2. 1

[8] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, Sept. 1997. 9

[9] C. Feng, Y. Taguchi, and V. R. Kamat. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6225, May 2014. ISSN: 1050-4729. 2

[10] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 3

[11] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24*, pages 1–11, July 2024. arXiv:2403.17888 [cs]. 1

[12] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. Large Scale Multi-view Stereopsis Evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413, June 2014. ISSN: 1063-6919. 2, 5, 6, 8, 9

[13] L. Jin, S. Qian, A. Owens, and D. F. Fouhey. Planar Surface Reconstruction from Sparse Views. In *arXiv:2103.14644 [cs]*, Mar. 2021. arXiv: 2103.14644. 3

[14] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. *arXiv:1711.07566 [cs]*, Nov. 2017. arXiv: 1711.07566. 2

[15] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, Aug. 2017. 2, 7

[16] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6):1–14, Dec. 2020. 5

[17] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin. Neuralangelo: High-Fidelity Neural Surface Reconstruction. pages 8456–8465, 2023. 1, 4

[18] S. Liu, T. Li, W. Chen, and H. Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *arXiv:1904.01786 [cs]*, Apr. 2019. arXiv: 1904.01786. 2

[19] S. Liu, Y. Yu, R. Pautrat, M. Pollefeys, and V. Larsson. 3D Line Mapping Revisited, Mar. 2023. arXiv:2303.17504 [cs]. 2

[20] W. Liu, Y. Wu, S. Ruan, and G. S. Chirikjian. Robust and Accurate Superquadric Recovery: A Probabilistic Approach. pages 2676–2685, 2022. 3

[21] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, Aug. 1987. 2, 3, 4, 9

[22] I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts, May 2017. arXiv:1608.03983 [cs, math]. 5

[23] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization, Jan. 2019. arXiv:1711.05101 [cs, math]. 6

[24] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *arXiv:1812.03828 [cs]*, Apr. 2019. arXiv: 1812.03828. 2

[25] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *arXiv:2003.08934 [cs]*, Aug. 2020. arXiv: 2003.08934. 1, 2, 3, 4, 5

[26] N. Mitra, M. Wand, H. R. Zhang, D. Cohen-Or, V. Kim, and Q.-X. Huang. Structure-aware shape processing. In *SIGGRAPH Asia 2013 Courses*, SA '13, pages 1–20, New York, NY, USA, Nov. 2013. Association for Computing Machinery. 1

[27] T. Monnier, J. Austin, A. Kanazawa, A. A. Efros, and M. Aubry. Differentiable blocks world: Qualitative 3d decomposition by rendering primitives. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 2, 3, 5, 6, 7

[28] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. Publisher: ACM New York, NY, USA. 3, 5, 8

[29] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. *arXiv:1912.07372 [cs, eess]*, Mar. 2020. arXiv: 1912.07372. 2

[30] M. Oechsle, S. Peng, and A. Geiger. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. *arXiv:2104.10078 [cs]*, Apr. 2021. arXiv: 2104.10078. 2, 3

[31] D. Paschalidou, A. O. Ulusoy, and A. Geiger. Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids. pages 10344–10353, 2019. 5

[32] J. Philip and V. Deschaintre. Floaters No More: Radiance Field Gradient Scaling for Improved Near-Camera Training, June 2023. arXiv:2305.02756 [cs]. 8

[33] M. Ramamonjisoa, S. Stekovic, and V. Lepetit. MonteBoxFinder: Detecting and Filtering Primitives to Fit a Noisy Point Cloud, July 2022. arXiv:2207.14268 [cs]. 3

[34] R. Ramamoorthi. NeRFs: The Search for the Best 3D Representation, Aug. 2023. arXiv:2308.02751 [cs]. 1

[35] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. SAM 2: Segment Anything in Images and Videos, Aug. 2024. arXiv:2408.00714 [cs]. 5

[36] L. G. Roberts. *Machine perception of three-dimensional solids*. Thesis, Massachusetts Institute of Technology, 1963. Accepted: 2005-08-17T19:45:17Z. 1

[37] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, Nov. 2008. 4

[38] J. L. Schonberger and J.-M. Frahm. Structure-from-Motion Revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, Las Vegas, NV, USA, June 2016. IEEE. 2, 4, 5, 9

[39] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys. Pixelwise View Selection for Unstructured Multi-View Stereo. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, volume 9907, pages 501–518. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computer Science. 2

[40] C.-Y. Sun, Q.-F. Zou, X. Tong, and Y. Liu. Learning adaptive hierarchical cuboid abstractions of 3D shape collections. *ACM Transactions on Graphics*, 38(6):1–13, Dec. 2019. 3

[41] J. Tang. nerf2mesh: Delicate textured mesh recovery from nerf via adaptive surface refinement. https://github.com/ashawkey/nerf2mesh, 2023. Accessed 2024-10-07. 7

[42] J. Tang, H. Zhou, X. Chen, T. Hu, E. Ding, J. Wang, and G. Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 17693–17703. IEEE, 2023. 5, 6, 7, 8, 9

[43] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik. Learning Shape Abstractions by Assembling Volumetric Primitives. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1466–1474, Honolulu, HI, July 2017. IEEE. 3

[44] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27171–27183, 2021. 1, 2, 3

[45] T. Wu, J. Wang, X. Pan, X. Xu, C. Theobalt, Z. Liu, and D. Lin. Voxurf: Voxel-based Efficient and Accurate Neural Surface Reconstruction, Apr. 2023. arXiv:2208.12697 [cs]. 1

[46] Y. Wu, W. Liu, S. Ruan, and G. S. Chirikjian. Primitive-based Shape Abstraction via Nonparametric Bayesian Inference, July 2022. arXiv:2203.14714 [cs]. 2, 3

[47] Y. Xie, M. Gadelha, F. Yang, X. Zhou, and H. Jiang. PlanarRecon: Real-time 3D Plane Detection and Reconstruction from Posed Monocular Videos, June 2022. arXiv:2206.07710 [cs]. 3

[48] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan. BlendedMVS: A Large-Scale Dataset for Generalized Multi-View Stereo Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1787–1796, Seattle, WA, USA, June 2020. IEEE. 2, 7

[49] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman. Volume Rendering of Neural Implicit Surfaces. *arXiv:2106.12052 [cs]*, June 2021. arXiv: 2106.12052. 1, 2, 3, 4

[50] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, R. Basri, and Y. Lipman. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. *arXiv:2003.09852 [cs]*, Oct. 2020. arXiv: 2003.09852. 2, 3

[51] F. Yu, Y. Qian, X. Zhang, F. Gil-Ureta, B. Jackson, E. Bennett, and H. Zhang. DPA-Net: Structured 3D Abstraction from Sparse Views via Differentiable Primitive Assembly, Aug. 2024. arXiv:2404.00875 [cs]. 3