Differentiable Convex Polyhedra Optimization from Multi-view Images

Daxuan Ren[1], Haiyi Mei,[2], Hezi Shi[1],
Jianmin Zheng✉[1], Jianfei Cai[1,3], and Lei Yang[2]

[1] College of Computing & Data Science, Nanyang Technological University, Singapore
[2] Sensetime Research
[3] Department of Data Science & AI, Monash University Monash University
{daxuan001, hezi001}@e.ntu.edu.sg, {meihaiyi, yanglei}@sensetime.com,
asjmzheng@ntu.edu.sg, jianfei.cai@monash.edu

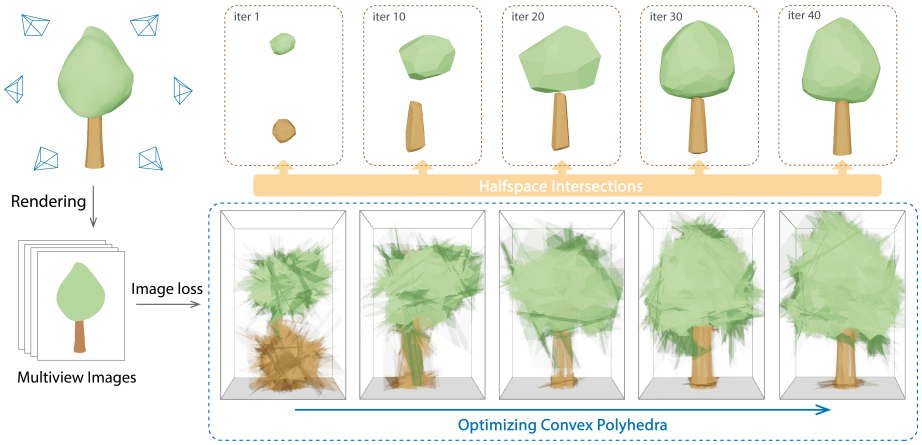

Fig. 1: A novel method optimizes differentiable convex polyhedra w.r.t image losses, bridging the gap between compact shape representation and easily obtained image supervision.

Abstract. This paper presents a novel approach for the differentiable rendering of convex polyhedra, addressing the limitations of recent methods that rely on implicit field supervision. Our technique introduces a strategy that combines non-differentiable computation of hyperplane intersection through duality transform with differentiable optimization for vertex positioning with three-plane intersection, enabling gradient-based optimization without the need for 3D implicit fields. This allows for efficient shape representation across a range of applications, from shape parsing to compact mesh reconstruction. This work not only overcomes the challenges of previous approaches but also sets a new standard for representing shapes with convex polyhedra.

1 Introduction

The quest for shape representation in computer graphics and computer vision has led to significant advancements, exploring various paradigms such as point clouds, voxels, meshes, implicit fields, and geometric primitives. Among these, the technique of using a combination of simple geometric primitives to represent more complex shapes is a promising way of capturing the geometric essence of objects [4,7,33,39,40,43,53,54]. Particularly, representing shapes via a set of convex polyhedra has shown potential in many downstream applications [19,20], including parsing shapes to semantically meaningful pieces, efficient physics-based simulations, collision detection, and compact mesh representations for efficient rendering and storage. Despite its potential, a fundamental challenge remains: how to enable the differentiable construction and optimization of convex polyhedra. This capability is crucial for leveraging gradient-based optimization techniques, thus enhancing the adaptability and precision of shape representations within learning-based frameworks.

Recent advances in shape representation with primitives have predominantly relied on supervision from implicit fields [4,7,13,16,39,40,53,54], such as Signed Distance Functions (SDF) or occupancy fields. While effective, these methods necessitate watertight meshes for training, which can be difficult to acquire. Apart from this, preprocessing the dataset into a set of discrete sample points with ground truth implicit values imposes a huge burden on both computation resources and storage. Additionally, during training, all of these methods define shape compositing operations such as intersecting planes and uniting convex polyhedra in approximated ways, like sigmoid for converting SDF into occupancy, and softmin and softmax for intersection and union, leading to inaccurate overall SDF or occupancy fields or biased gradient estimate. Also, since the gradient of each sample point w.r.t all the primitives needs to be cached during training, the consumption of GPU memory also imposes a barrier for scaling up the models.

This paper presents an approach that provides a new direction for eliminating the dependency on implicit fields. It is grounded on purely explicit surface models and leverages images rendered in a differentiable manner for supervision. This strategy offers a more flexible solution for gradient-based optimization of shape representation through a collection of convex polyhedra.

Specifically, our method combines highly efficient non-differentiable computations with the adaptability of differentiable operations. Utilizing the duality transform concept [6, 34], we precisely identify the hyperplanes that form each vertex of a convex polyhedron—an inherently non-differentiable task. With the hyperplane intersections mapped to each vertex of the polyhedron, we then employ a differentiable linear system solver to calculate the vertex positions by solving for the intersection of three planes. This approach ensures the seamless backpropagation of gradients from image-based losses to the plane parameters defining the convex polyhedron, making the whole optimization process differentiable. Since our method only requires images as training data, it removes the

requirements for ground truth meshes, unlocking training data with orders of magnitude larger size.

In our approach, each vertex location is directly solved explicitly as three-plane intersection, which makes the reconstructed mesh much more accurate in representing the shape compared to previous methods that leverage implicit fields, and offers more accurate gradients around shape boundaries as well.

Consequently, our method paves the way for more efficient and scalable shape representation learning, making it particularly advantageous for applications where high fidelity is required.

We showcase the broad applicability and superior effectiveness of our method through its successful deployment in a diverse array of applications. Our technique excels in shape reconstruction, where it achieves high-fidelity representations of complex geometries, especially on parts with detailed geometry. In textured multiview reconstruction, it seamlessly integrates diverse visual perspectives into coherent, detailed models, underscoring its ability to handle varied data inputs. Additionally, our approach demonstrates its strength in shape parsing by accurately decomposing complex shapes into semantically meaningful components. All these applications illustrate not only the versatility of our method but also its capacity to deliver enhanced performance and efficiency.

In summary, this paper has the following contributions:

- We introduce a novel method for making the construction of convex polyhedra differentiable, enabling gradient-based optimization without reliance on implicit fields.
- Our approach combines non-differentiable computation via duality transform with differentiable optimization, allowing for effective and accurate shape representation.
- We demonstrate the versatility and efficiency of our method across a range of applications, from shape reconstruction, textured multiview reconstruction, to shape parsing.

2 Related Work

This section briefly discusses related topics, focusing on shape representations, especially with geometric primitives, and differentiable rendering.

2.1 Shape Representations

Voxels directly expand pixels from 2D images into 3D space, representing shapes with a regular grid of cubic cells (voxels). Each cell can store information such as material properties, occupancy, or color. This representation is well-suited for volumetric data, offering straightforward manipulation and visualization. However, it is often limited by high memory consumption [23, 31, 36, 47].

Point clouds represent a shape as a collection of discrete points in space, capturing the surface geometry without explicit connectivity information. This representation is widely used in 3D scanning and reconstruction, where raw data

typically form a point cloud. Despite its popularity, the lack of surface connectivity requires additional processing for applications that demand explicit surface models. Recently, many methods have been proposed for various tasks based on point clouds, ranging from shape understanding [35, 37, 48], shape reconstruction [51] to shape generation [28].

Meshes use vertices, edges, and faces in a connected topology to represent shapes. Triangular and quadrilateral meshes are the most common, offering detailed and flexible representations of complex surfaces. Meshes are a cornerstone in modern computer graphics for modeling, rendering, animation, and geometric analysis [5, 8, 29, 45, 49].

Implicit fields (or implicit surfaces) define shapes through scalar fields, with the surfaces represented as level sets of the implicit functions (e.g., the set of points whose signed distance function values are zero). This representation supports complex topologies and smooth surfaces, and is also favored for blending and modeling smooth transitions between shapes. In recent years, leveraging implicit fields has become a hot topic for its capability of representing smooth shapes with complicated topology and easy integration into deep learning frameworks. Implicit fields have been widely used for shape learning [9, 24, 32] and differentiable rendering in computer vision [25, 46, 50, 52, 55].

Hybrid Approaches represent shapes using both an implicit field and a surface mesh via differentiable iso-surface extraction methods. For instance, Deep Marching Cubes [18] adapts the classic marching cubes algorithm for differentiable use, enabling the extraction of surface geometry from volumetric representations through learnable parameters. Deep Marching Tetrahedra [41] further extends this approach by introducing differentiability into the process of converting scalar fields to meshes via deformable tetrahedron grids. This facilitates the optimization of complex geometries using gradient descent. FlexiCube [42] proposes a differentiable iso-surface extraction method based on Dual Marching Cubes, offering efficient and flexible shape optimization.

Primitive-based representations model shapes by combining basic geometric entities such as planes, spheres, cubes, cylinders, quadrics, superqudrics [33, 43]. While they might not be as flexible or might not capture as much detail as other representations, they offer advantages on shape parsing, compactness, editability, etc. One prominent representation is Constructive Solid Geometry (CSG) [4, 7, 16, 39, 40, 53, 54], which models complex objects by combining simple solid primitives using Boolean operations (union, intersection, difference). Among the existing methods, CVXNet [7] and BSPNet [4] are most similar to ours. Both of them represent shapes implicitly as the union of a set of convex polyhedra, which result from intersecting hyperplanes.

2.2 Differentiable Rendering

Differentiable rendering bridges the gap between 3D model representation and image-based loss functions, allowing for the optimization of shape parameters directly from image data. Differentiable rendering can be roughly categorized into three categories: rasterization-based, physical-based, and implicit field based.

Rasterization based differentiable renderers [15,21,27,38] generally offer fast rendering, which use a "soft" version of rasterization and shading to allow gradients to propagate, however it generally only support explicit triangle mesh as shape representation. Unlike rasterization based differentiable renderers, physical-based differentiable renderers are built on top of ray tracing techniques and support a border spectrum of light transport phenomena such as reflection, refraction, and transparent objects [2,10,17,22,30], these methods can support shape representation other than triangle mesh, as long as explicit boundaries or a continuous and boundary consistent reparameterization is provided. With the research advancement in the field of implicit fields, the renders for implicit fields [1,12,25,44,46,52] also become popular. Typically, these methods use volume rendering to integrate radiance along the ray direction and employ alpha compositing to determine each pixel's color. Additionally, 3DGS [14] proposed a hybrid approach that represents scenes with gaussian balls, combining rasterization with volumetric rendering to generate images. These methods have recently gained significant attention due to the flexibility and continuous nature of implicit fields. However, none of these method can be applied directly to in our case, since: **1)** there is no explicit surface mesh available, **2)** a continuous and boundary-consistent reparameterization is hard to design, especially with potential self-intersecting convex polyhedra, and **3)** volume rendering require untractable amount of VRAM when paired with the primitive-based representations discussed in Sec.2.1. For a 512x512 image with 64 convex polyhedra, each defined by 64 hyperplanes, and each ray sampling 512 points for ray integration, the total number of queries per image becomes impractically high (512x512x64x64x512 = 549,755,813,888). This exceeds the capacity of contemporary GPU memory. These challenges motivated the approach described in this work.

3 Methods

Overview. The essence of our approach is to employ the intersections and unions of a set of halfspaces to depict shapes. To equip these shapes with differentiability, we initially construct convex polyhedra from the intersections of halfspaces using non-differentiable duality transforms and record the IDs of three planes that intersect to form each vertex. Subsequently, we recompute the vertex positions of polyhedra through a differentiable process for finding the intersection of three planes. This enables the vertex positions to be differentiable, allowing the straightforward application of a differentiable renderer to calculate image loss and facilitating the backpropagation of gradients to the plane parameters. Notably, the union of convex polyhedra is seamlessly managed by the renderer, utilizing mechanisms such as the z-buffer or ray-object intersections, thereby obviating the need for explicit handling. Fig. 2 illustrates these processes.

3.1 Differentiable Rendering of Convex Polyhedra

Differentiable rendering of convex polyhedra involves the computation of gradients of plane parameters θ w.r.t image loss $\mathcal{L}$: $\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \mathcal{I}} \cdot \frac{\partial \mathcal{I}}{\partial \theta}$.

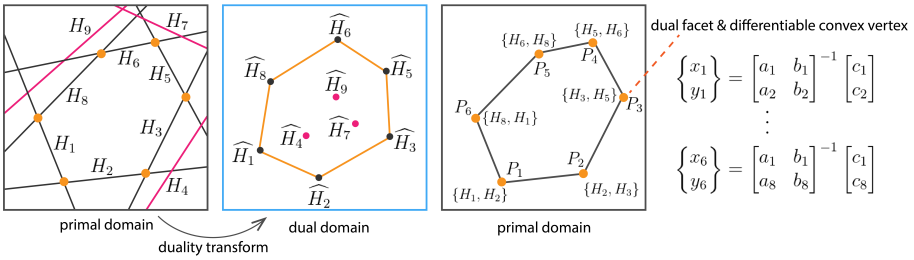

Fig. 2: Overview of the proposed method. Given a set of hyperplanes, we use duality transform to map them into the dual space. We then compute the convex hull of the dual vertices. Each facet of the dual convex hull represent a intersection point in the primal domain. Once the plane IDs for each intersection vertex are recorded, we can recompute the vertex location via differentiable linear equation solvers.

However, directly evaluating $\frac{\partial \mathcal{I}}{\partial \theta}$ is difficult, see Sec.2.2. Thus, we choose an indirect approach by constructing the mesh of the convex polyhedron where the vertex position is differentiable w.r.t plane parameters. After this, we can use standard differentiable renderer to obtain the final gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathcal{I}}}_{\text{Differentiable Image Loss}} \cdot \underbrace{\frac{\partial \mathcal{I}}{\partial \mathcal{V}}}_{\text{Differentiable Rendering}} \cdot \underbrace{\frac{\partial \mathcal{V}}{\partial \theta}}_{\text{Differentiable Vertex Location}} \tag{1}$$

3.2 Halfspace Intersections about a Point

In Euclidean space $\mathbb{R}^n$, a halfspace is the set of points $x \in \mathbb{R}^n$ satisfying a linear inequality of the form $a^T x \leq b$, where $a \in \mathbb{R}^n$ is a nonzero vector, $b \in \mathbb{R}$ is a scalar, and $a^T x$ represents the dot product of a and x.

A convex set (polyhedron) is a subset of $\mathbb{R}^n$ that, for every pair of its points, the line segment connecting the two points is contained in the set. Formally, a set $C \subseteq \mathbb{R}^n$ is convex if for every $x, y \in C$ and every $\lambda \in [0, 1]$, the point $\lambda x + (1 - \lambda)y$ is also in C.

The convex set generated by intersecting a set of halfspaces is defined as follows. Let $\{H_i\}_{i=1}^k$ be a collection of halfspaces in $\mathbb{R}^n$, where each halfspace H_i is defined by a linear inequality $a_i^T x \leq b_i$ with $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. The intersection of these halfspaces, denoted by C, is defined as:

$$C = \bigcap_{i=1}^{k} H_{=} \left\{ x \in \mathbb{R}^n \mid a_i^T x \leq b_i, \forall i = 1, \ldots, k \right\}. \tag{2}$$

Duality Transform. The duality transform is a powerful tool in computational geometry that allows us to view the problem from a different perspective. To give a concrete example, a line in 2D Euclidean space has two parameters, i.e., the slope m and the Y-intercept n. Therefore, we can draw a point in the dual domain with coordinate (m, n). This mapping between the line in the primal

domain and the point in the dual domain (and vice versa) is known as the duality transform. Duality transform can be very useful in many computation geometry applications, such as the construction of convex hull, Delaunay triangulation, Voronoi diagram, and in our case, halfspace intersections.

Halfspace Intersection with Duality Transform. Given a set of halfspaces defined by linear inequalities $a_i^T x \leq b_i$ for $i = 1, \ldots, k$, and a feasible point x_0 that satisfies all these inequalities, the duality transform involves converting each halfspace into a point in the dual domain and vice-versa. This transformation facilitates the identification of all vertices generated by intersecting the halfspaces. Specifically, the dual of a hyperplane $a_i^T x = b_i$ is defined by a point with coordinates $(\frac{a_{i_x}}{b_i}, \frac{a_{i_y}}{b_i}, \frac{a_{i_z}}{b_i})$ in the dual domain. By computing the convex hull of these points in the dual domain, we can effectively identify the boundaries of the intersection of halfspaces in the primal domain. Notably, each facet of the dual convex hull corresponds to a vertex in the primal domain formed by intersecting the hyperplanes corresponding to the vertices of the dual facet [34]. This approach can significantly simplify the computation by converting a potentially complex halfspace intersection problem into a convex hull problem in the dual domain. Once the three planes that intersect into each convex polyhedron vertex is know, the position of the vertex can be computed differentially via the solution of a system of linear equations. Specifically, if a vertex in the primal space is formed by the intersection of n hyperplanes, then the position of this vertex, denoted by x, can be found by solving the system: $Ax = b$ where A is a matrix whose rows are the normal vectors of the hyperplanes a_i^T for $i = 1, \ldots, n$, and b is a vector containing the offsets of these hyperplanes, with each entry b_i corresponding to the distance from the origin to the hyperplane. Formally, this system can be written as:

$$\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} x = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}. \tag{3}$$

The solution to this system gives the coordinates of the vertex in the primal space. Since solving the linear system can be done in a differentiable fashion, $\frac{\partial \mathcal{V}}{\partial \theta}$ can be obtained.

Having tackled vertex computation, we shift our attention to triangle face generation. As optimizing the connectivity of vertices often falls outside the scope of most differentiable renderers, we opt for directly utilizing the facets of the convex hull constructed in the primal space through non-differentiable methods and tessellate them into triangles. After computing the vertices and triangle faces of a convex polyhedron, the mesh is fed into a differentiable renderer for rendering.

3.3 Optimization Strategies

Having discussed the construction of an individual convex polyhedron, now we turn to optimizing a set of convex polyhedra to represent a scene with com-

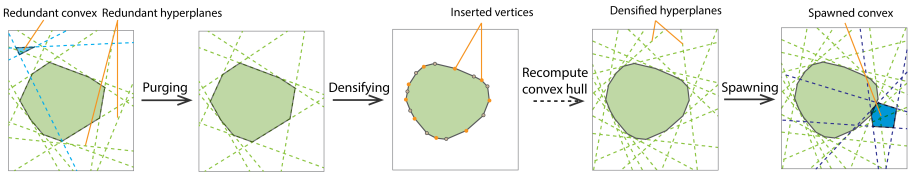

Fig. 3: Small convex polyhedra and redundant planes will be removed to speed up the optimization process. To better reconstruct the shape with high curvature, we employ a densification process that constructs the mesh of the convex polyhedron, runs Loop subdivision, and uses the recomputed convex hull equations of the subdivided mesh to serve as the updated plane parameters.

plex geometries. Inspired by 3DGS [14], we adapted several heuristics during optimization. Fig.3 illustrates an overview of operations on convex polyhedra.

Convex Initialization. We commence our scene construction with a predetermined quantity of convex polyhedra and planes. The initialization process assigns a uniform size to all convex polyhedra, denoted by the same b value as outlined in Eq. 2. Convexes are randomly initialized throughout the space.

Persistent Convex. To facilitate the intersection of hyperplanes through duality, our method begins by securing a feasible starting point. Although identifying a feasible solution such as determining the Chebyshev center of a linear programming problem is relatively straightforward, we circumvent this necessity by ensuring that the origin is always within the convex hull. This is accomplished by introducing additional constraints to Eq. 2, mandating that all plane offsets remain positive ($b \in \mathbb{R}^+$). This strategy ensures that the origin acts as a feasible solution. Moreover, to provide spatial flexibility, we incorporate an extra translation parameter for each convex. This adjustment allows for the seamless relocation of the entire convex polyhedron within the three-dimensional space, enhancing the adaptability of our approach.

Convex Purging. The treatment to convex polyhedra introduced above guarantees their persistence by consistently enclosing a viable region, thus preventing from the elimination of small convex polyhedra throughout the optimization process. To improve both the visual quality and computational efficiency, we adopt a convex purging strategy. This strategy entails the removal of convex polyhedra along with their associated parameters once their volumes are below a certain predefined threshold.

Plane Purging. During optimization, if a plane does not contribute to the computation of any vertex, it will not receive any gradient and never get updated. Thus, we further prune inactive hyperplanes to expedite the optimization. This can be done by removing all the hyperplanes that have a dual vertex falling within the dual convex hull.

Convex Densification. Given the initialization of the convex polyhedron with a relatively small number of planes, our approach initially struggles to accurately reconstruct regions exhibiting high curvature. To address this limitation, we employ a densification strategy that reintroduces the necessary hyperplanes, and

provides a balance between model sparsity and the need for detailed shape reconstruction. Specifically, after a predetermined number of optimization iterations, we activate a convex densification process aimed at enhancing the reconstruction of areas with high curvature. This process is executed through a method akin to mesh subdivision: for each convex polyhedron, we calculate the triangle mesh of the convex polyhedron, apply one Loop subdivision iteration, and then recompute the convex hull of the subdivided mesh. The plane equations derived from the new convex hull are then utilized as the parameters for the planes in the subseqent optimization iterations, ensuring a more accurate and comprehensive representation of complex shapes.

Convex Spwaning. Convex purging, the process of discarding small convex polyhedra, potentially risks entrapment in local minima and the loss of details. To counteract this and ensure that a sufficient number of convex polyhedra are employed for the final shape reconstruction, we introduce a convex spawning operation. This procedure involves randomly re-initializing removed convex polyhedra within the space. Through practical applications, we have found that this random spawning approach is effective in maintaining an adequate count of convexes for the final reconstruction. Nonetheless, we acknowledge the possibility of more sophisticated spawning strategies that could enhance this process, suggesting a direction for future research.

4 Experiments

4.1 Shape Reconstruction

Dataset We begin experiments with multiview reconstruction using the ShapeNet dataset [3]. Our study focuses on a standard subset of 13 categories. Given that our method does not include a learning component, we evaluate our approach on 100 randomly selected shapes from each category, constrained by computational resource limitations. For each shape, we render 16 images from different viewpoints to act as supervision data.

Implementation Details We implement our method using C++ with QHull for duality related computation. To facility gradient base optimization, we implemented a Python wrapper for it. Once the plane IDs for each convex polyhedron vertex is outputted, we use PyTorch's linear algebra solver to determine the vertex position. We use a total of 32 convex polyhedra for our experiment. We randomly initialize the convex polyhedra in space and proceed the optimization steps as detailed in Sec.3.3. We optimize a total of 20000 steps, incorporating 10 densification and random spawning steps to refine our model further. We use NVDiffRast [15] as our renderer and build our pipeline based on [22], due to the code readability and ease of integration. Note that our implementation does not receive any specialized treatments in [22] such as re-parameterization. Instead, we simply reuse their rendering setup such as lighting and shading. Our pipeline is supervised by purely image L_1 loss. We use a learning rate of $1e^{-2}$ for convex polyhedron translation and $1e^{-3}$ for hyperplanes.

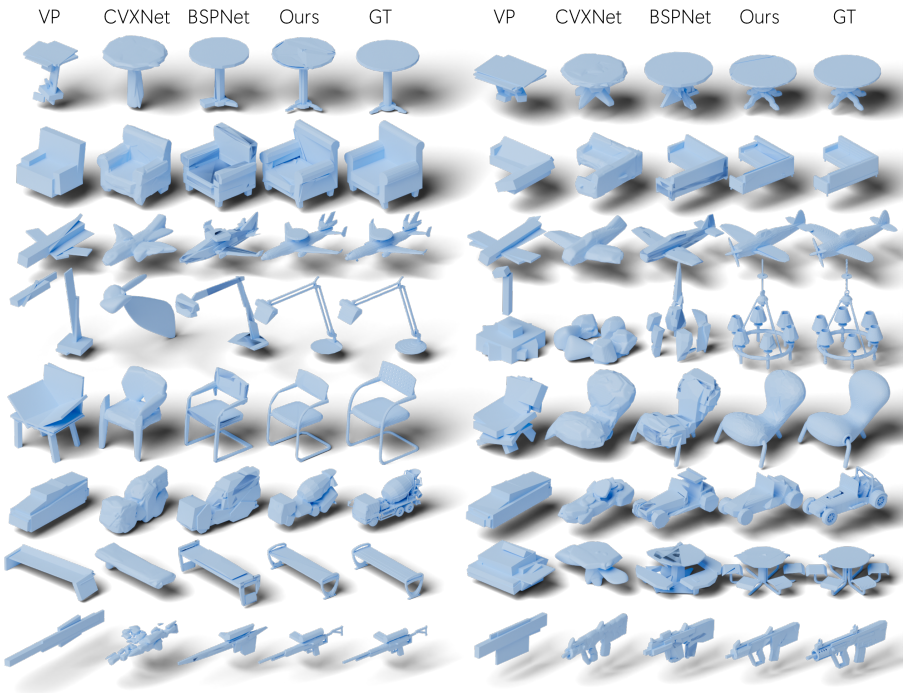

Fig. 4: Comparing our method with VP, CVXNet, and BSPNet on the ShapeNet dataset. The visualization results show that our method generates better reconstruction, especially in the thin and detailed aspects of shapes.

Comparisons In our evaluation, we benchmark our method against other methods focused on shape representation through basic geometric primitives, such as VP [43]with 32 cuboids (increased compare to their original paper), CVXNet [7] with 64 convexes (same as their paper), and BSPNet [4] with 64 convexes (same as their paper). Given that our approach does not have a "learning" component and is designed to optimize convex parameters to "overfit" to individual shapes, the metric of the baseline method is also "overfit" to the dataset, i.e. using training set as test set. Acknowledging our method's lack of visual supervision for internal structures, we employ a postprocessing step from OccNet [24] that sets random cameras, renders multiple depth image and fuses them into a final surface mesh. We use this method to effectively remove any internal structure from both ground truth and output shapes, facilitating fair comparisons.

In Tab.1, we detail our evaluation through quantitative metrics, i.e., L_1 Chamfer Distance, L_2 Chamfer Distance (multiplied by 1000 for better viewing), and Normal Consistency. Our method outperforms the baselines on most benchmarks. Qualitative comparisons are shown in Fig.4. These assessments demonstrate that our method excels in the precise reconstruction of thin and intricate details, for example, the Lamp category in Fig.4. However, we observe

Table 1: Comparison of reconstruction results with baselines, measured by L_1 Chamfer Distance, L_2 Chamfer Distance, and Normal Consistency. Our method outperforms the baselines and achieves the best overall reconstruction results.

	L_1 Chamfer Distance				L_2 Chamfer Distance x1000				Normal Consistency			
	VP	CVX	BSP	Ours	VP	CVX	BSP	Ours	VP	CVX	BSP	Ours
plane	0.036	0.023	0.017	**0.011**	0.997	0.419	0.242	**0.084**	0.709	0.836	0.791	**0.958**
car	0.054	0.023	0.031	**0.019**	2.157	**0.346**	0.739	0.363	0.729	0.877	0.675	**0.937**
chair	0.052	0.024	0.027	**0.023**	2.130	0.769	**0.546**	0.778	0.694	0.932	0.751	**0.932**
lamp	0.056	0.023	0.032	**0.020**	2.747	**0.488**	1.493	0.823	0.642	0.852	0.692	**0.932**
table	0.046	**0.023**	0.026	**0.023**	1.690	**0.449**	0.604	0.510	0.841	**0.938**	0.803	**0.938**
sofa	0.057	**0.021**	0.029	0.023	2.272	**0.283**	0.675	0.442	0.677	**0.954**	0.731	0.920
phone	0.043	**0.019**	0.025	0.022	1.382	**0.216**	0.446	0.346	0.882	**0.966**	0.820	0.962
vessel	0.045	0.028	0.025	**0.015**	1.571	0.687	0.558	**0.382**	0.679	0.802	0.707	**0.935**
speaker	0.060	**0.027**	0.041	0.038	2.696	**0.591**	1.564	1.482	0.730	**0.944**	0.710	0.869
cabinet	0.057	**0.026**	0.036	0.042	2.461	**0.440**	1.095	1.601	0.730	**0.947**	0.742	0.846
display	0.046	**0.021**	0.025	0.026	1.553	**0.271**	0.516	0.557	0.859	**0.963**	0.821	0.937
bench	0.043	0.021	0.022	**0.016**	1.513	0.371	0.389	**0.254**	0.716	0.870	0.725	**0.936**
rifle	0.034	0.036	0.016	**0.010**	1.022	1.514	0.215	**0.070**	0.712	0.700	0.714	**0.926**
mean	0.048	0.024	0.027	**0.022**	1.861	**0.526**	0.699	0.592	0.738	0.891	0.745	**0.925**

performance discrepancies between L_1 and L_2 Chamfer Distances, caused by some floating convexes that have not been removed by the optimization. Additionally, for categories with large cavities (e.g., cabinets), our method does not perform as well, likely due to occlusion when relying solely on RGB image supervision.

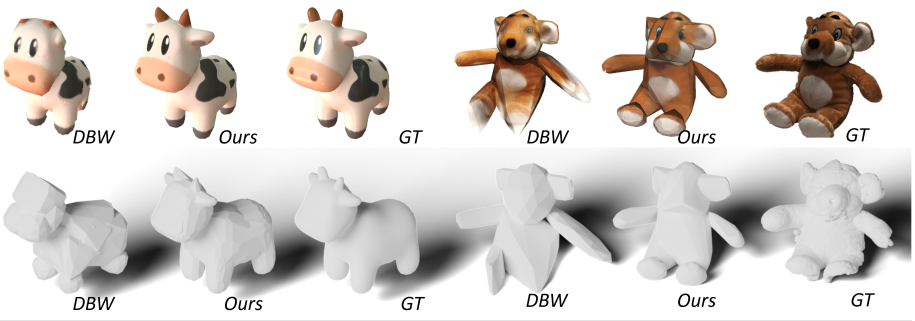

Fig. 5: We assess our convex polyhedron-based method against DBW [26] that is based on boxes and superquadrics. Visual comparisons demonstrate that our approach yields much better reconstruction results.

4.2 Multiview Reconstruction

In this experiment, we extend our model to real world examples, showcasing its effectiveness across various settings and datasets. Given the nature of convex polyhedra, which lack consistent tessellation and triangle topology for a static texture mapping, we adapt the technique from NVDiffRec [27], applying

volumetric textures while solely altering the geometry representation. Our evaluations span both synthetic and real-world captured datasets. We benchmark our approach against models based on other geometric primitives, i.e. DifferentiableBlockWorlds [26]. We run our method with DTU datasets [11] as well as other synthetic datasets with textures. The quantitative metric of [26] on DTU dataset reported in Tab.2 is from the original paper, and the metric of our method is generated with the same evaluation script from [26]. For synthetic datasets (e.g., NeRF Datasets [25]), we only adopt the superquadric geometry from DBW and use the same pipeline as NVDiffRec [27], and the metric is computed using L_1 Chamfer Distance (multiplied by 10 for better viewing). We show qualitative comparison in Fig. 5, which demonstrates better geometry and overall reconstruction quality of our method.

Table 2: Comparison of our method with the method of Differentiable Block Worlds (DBW) [26] for multiview reconstruction. We can see that our method outperforms DBW by a large margin.

		DTU					Mesh		NeRF Synthetic				
	Method	S24	S40	S55	S83	S105	Bob	Spot	Lego	Chair	Mic	Drums	Hotdog
L_2 CD	**DBW [26]**	**3.25**	1.16	2.98	3.43	5.21	0.71	1.03	0.91	1.07	1.06	1.07	1.40
L_2 CD	**Ours**	3.87	**1.01**	**2.43**	**2.49**	**2.62**	**0.43**	**0.65**	**0.44**	**0.35**	**0.25**	**0.41**	**0.63**

4.3 Shape Parsing

We have demonstrated the reconstruction ability of our method. Now we delve into a detailed analysis of our reconstruction results, focusing particularly on shape parsing and segmentation capabilities of our method. Our examples span both CAD model types, such as rifles and airplanes, and organic shapes like bunnies and bobs. Fig. 6 displays the wire frames of the reconstructed meshes. We can see from the figure that the output mesh is dense in the region with geometric details and sparse in flat regions. We also color-code individual convex polyhedra of the reconstructed shapes, as shown in Fig. 7. Additionally, we manually group the convexes to illustrate their correspondence with specific parts of the objects, further highlighting our method's adeptness in understanding and segmenting complex shapes.

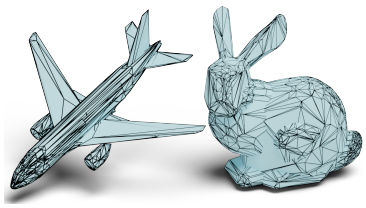

Fig. 6: Wireframes of the extracted meshes show the compactness of our outputs.

4.4 Ablation Study

An ablation study is conducted to understand the impact of various settings on the performance of our method. Specifically, we explore the effects of convex den-

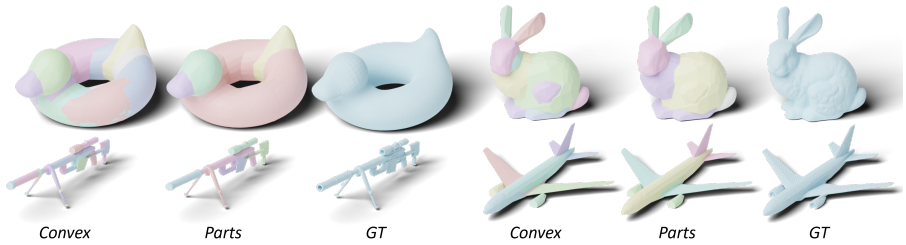

Fig. 7: Four objects: Bob, Bunny, Bench, and Airplane. For each object, we show the color-coded individual convex polyhedra on the left and manually grouped object parts in the middle.

sification, convex spawning and the number of convexes utilized. Each of these components can play a role in the overall effectiveness and efficiency of shape reconstruction, and understanding their individual and combined contributions is essential for refining the approach. We discuss the effect of each component in the following sections and show numerical results in Tab.3

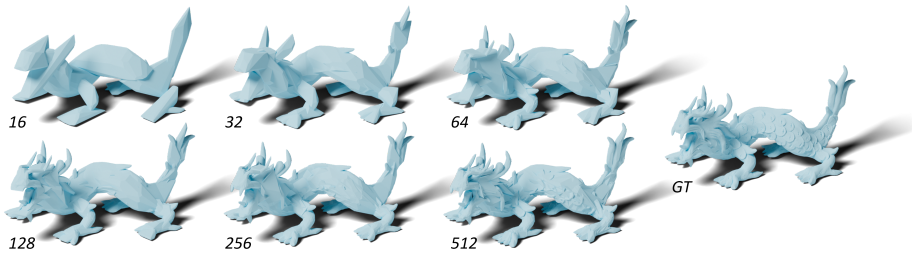

Fig. 8: Reconstruction using different numbers of convexes. A smaller number of available convex polyhedra leads to a higher abstraction level with a more prominent shape parsing structure, while a higher number leads to better reconstruction results, especially on the region with geometry details.

Number of Convexes The number of available convex polyhedra used in the model directly impacts its capacity to reconstruct detailed shapes. We conduct experiments by varying the number of convex polyhedra to visualize the balance between reconstruction accuracy and abstraction levels. Visual results are given in Fig. 8, where a complex shape (dragon) is optimized with different numbers of convex polyhedra. From the visualization we can see that the higher number gives

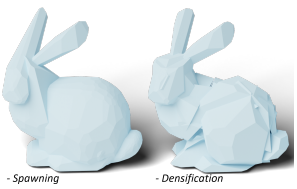

Fig. 9: Left without convex spawning, **Right** without densification process.

generally better reconstruction results but with the cost of less abstraction as well as more computation overhead.

	16 Convex	32 Convex	64 Convex	128 Convex	256 Convex	512 Convex	-Densify$_{16}$	-Spawn$_{16}$
CD L_2	0.32	0.14	0.08	0.05	0.05	0.05	0.46	0.54

Table 3: Quantitative results from different optimization configurations.

Densification As discussed in Sec. 3.3, we employed a process of adding additional hyperplanes to a convex polyhedron to better capture regions of high curvature or complex detail. We compare models optimized with and without the implementation of convex densification to evaluate its influence on the fidelity of the reconstructed shapes, as shown in Fig.9. We can see from the comparison that the densification process drastically enhances the model's ability to represent curved surfaces, i.e., the heap of the bunny.
Convex Spawning In order to maintain the pre-specified number of convexes throughout the optimization process, we re-introduce convex polyhedra during optimization. This process is essential, as poor initial convex polyhedron location can lead to under-reconstruction even when the initial number of convex is large. We refer the readers to Fig. 9 for visual comparison.

5 Conclusion and Limitation

We have presented a new method for making the optimization of convex polyhedra differentiable w.r.t rendering loss. The key idea is to leverage a non-differentiable duality transform to identify planes intersecting at individual convex polyhedron vertices, which enables the process of solving vertex positions differentiable through the solution of three-plane intersections. The extensive experimentations and detailed ablation studies have demonstrated the effectiveness of our method. Our work will also benefit the research community by offering a new avenue for exploring shape representation with convex polyhedra via differentiable rendering techniques.
Limitation Despite its effectiveness, our method has certain limitations. Representing shapes as sets of convex polyhedra results in a loss of detail compared to ordinary meshes and implicit representations. This limitation also restricts current experiments to individual objects rather than entire scenes. The non-static mesh topology prevents predefined parameterization (UV unwrapping) for surface textures, necessitating volumetric textures. Additionally, the densification and purging process is somewhat heuristic. Our method relies on differentiable mesh rendering, inheriting their limitations, such as the lack of gradients for implicit edges (from triangle self-intersections). This can slow down the optimization process and cause it to get stuck in a local minimum. While these limitations do not significantly impact overall reconstruction performance, they highlight areas for further exploration.

Acknowledgements This work is supported by MOE AcRF Tier 1 Grant of Singapore (RG12/22), and also by the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore. Daxuan Ren was also partially supported by Autodesk Singapore.

References

1. Bangaru, S.P., Gharbi, M., Luan, F., Li, T.M., Sunkavalli, K., Hasan, M., Bi, S., Xu, Z., Bernstein, G., Durand, F.: Differentiable rendering of neural sdfs through reparameterization. In: SIGGRAPH Asia 2022 Conference Papers. pp. 1–9 (2022)
2. Bangaru, S.P., Li, T.M., Durand, F.: Unbiased warped-area sampling for differentiable rendering. ACM Transactions on Graphics (TOG) **39**(6), 1–18 (2020)
3. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
4. Chen, Z., Tagliasacchi, A., Zhang, H.: Bsp-net: Generating compact meshes via binary space partitioning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 45–54 (2020)
5. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), `http://www.blender.org`
6. De Berg, M.: Computational geometry: algorithms and applications. Springer Science & Business Media (2000)
7. Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., Tagliasacchi, A.: Cvxnet: Learnable convex decomposition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 31–44 (2020)
8. Guo, K., Zou, D., Chen, X.: 3d mesh labeling via deep convolutional neural networks. ACM Transactions on Graphics (TOG) **35**(1), 1–12 (2015)
9. Hao, Z., Averbuch-Elor, H., Snavely, N., Belongie, S.: Dualsdf: Semantic shape manipulation using a two-level representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7631–7641 (2020)
10. Jakob, W., Speierer, S., Roussel, N., Nimier-David, M., Vicini, D., Zeltner, T., Nicolet, B., Crespo, M., Leroy, V., Zhang, Z.: Mitsuba 3 renderer (2022), https://mitsuba-renderer.org
11. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanæs, H.: Large scale multi-view stereopsis evaluation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 406–413 (2014)
12. Jiang, Y., Ji, D., Han, Z., Zwicker, M.: Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1251–1261 (2020)
13. Kania, K., Zieba, M., Kajdanowicz, T.: Ucsg-net–unsupervised discovering of constructive solid geometry tree. arXiv preprint arXiv:2006.09102 (2020)
14. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
15. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics **39**(6) (2020)

16. Li, P., Guo, J., Zhang, X., Yan, D.M.: Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16816–16826 (2023)
17. Li, T.M., Aittala, M., Durand, F., Lehtinen, J.: Differentiable monte carlo ray tracing through edge sampling. ACM Transactions on Graphics (TOG) **37**(6), 1–11 (2018)
18. Liao, Y., Donne, S., Geiger, A.: Deep marching cubes: Learning explicit surface representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2916–2925 (2018)
19. Lien, J.M., Amato, N.M.: Approximate convex decomposition. In: Proceedings of the twentieth annual symposium on Computational geometry. pp. 457–458 (2004)
20. Lien, J.M., Amato, N.M.: Approximate convex decomposition of polyhedra. In: Proceedings of the 2007 ACM symposium on Solid and physical modeling. pp. 121–131 (2007)
21. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. The IEEE International Conference on Computer Vision (ICCV) (Oct 2019)
22. Loubet, G., Holzschuch, N., Jakob, W.: Reparameterizing discontinuous integrands for differentiable rendering. ACM Transactions on Graphics (TOG) **38**(6), 1–14 (2019)
23. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 922–928. IEEE (2015)
24. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
25. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision. pp. 405–421. Springer (2020)
26. Monnier, T., Austin, J., Kanazawa, A., Efros, A.A., Aubry, M.: Differentiable Blocks World: Qualitative 3D Decomposition by Rendering Primitives. In: NeurIPS (2023)
27. Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting triangular 3d models, materials, and lighting from images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8280–8290 (2022)
28. Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-e: A system for generating 3d point clouds from complex prompts. arXiv preprint arXiv:2212.08751 (2022)
29. Nicolet, B., Jacobson, A., Jakob, W.: Large steps in inverse rendering of geometry. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **40**(6) (Dec 2021). `https://doi.org/10.1145/3478513.3480501`, `https://rgl.epfl.ch/publications/Nicolet2021Large`
30. Nimier-David, M., Vicini, D., Zeltner, T., Jakob, W.: Mitsuba 2: A retargetable forward and inverse renderer. ACM Transactions on Graphics (TOG) **38**(6), 1–17 (2019)
31. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE international conference on computer vision. pp. 1520–1528 (2015)

32. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019)
33. Paschalidou, D., Ulusoy, A.O., Geiger, A.: Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10344–10353 (2019)
34. Preparata, F.P., Shamos, M.I.: Computational geometry: an introduction. Springer Science & Business Media (2012)
35. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
36. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5648–5656 (2016)
37. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
38. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Accelerating 3d deep learning with pytorch3d. arXiv:2007.08501 (2020)
39. Ren, D., Zheng, J., Cai, J., Li, J., Jiang, H., Cai, Z., Zhang, J., Pan, L., Zhang, M., Zhao, H., et al.: Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. arXiv preprint arXiv:2108.11305 (2021)
40. Ren, D., Zheng, J., Cai, J., Li, J., Zhang, J.: Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. In: European Conference on Computer Vision. pp. 482–498. Springer (2022)
41. Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S.: Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. Advances in Neural Information Processing Systems **34**, 6087–6101 (2021)
42. Shen, T., Munkberg, J., Hasselgren, J., Yin, K., Wang, Z., Chen, W., Gojcic, Z., Fidler, S., Sharp, N., Gao, J.: Flexible isosurface extraction for gradient-based mesh optimization. ACM Transactions on Graphics (TOG) **42**(4), 1–16 (2023)
43. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2635–2643 (2017)
44. Vicini, D., Speierer, S., Jakob, W.: Differentiable signed distance function rendering. Transactions on Graphics (Proceedings of SIGGRAPH) **41**(4), 125:1–125:18 (Jul 2022). `https://doi.org/10.1145/3528223.3530139`
45. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 52–67 (2018)
46. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
47. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions On Graphics (TOG) **36**(4), 1–11 (2017)
48. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019)

49. Wen, C., Zhang, Y., Li, Z., Fu, Y.: Pixel2mesh++: Multi-view 3d mesh generation via deformation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1042–1051 (2019)
50. Wu, Q., Liu, X., Chen, Y., Li, K., Zheng, C., Cai, J., Zheng, J.: Object-compositional neural implicit surfaces. arXiv preprint arXiv:2207.09686 (2022)
51. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5438–5448 (2022)
52. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. Advances in Neural Information Processing Systems **34**, 4805–4815 (2021)
53. Yu, F., Chen, Q., Tanveer, M., Amiri, A.M., Zhang, H.: Dualcsg: Learning dual csg trees for general and compact cad modeling. arXiv preprint arXiv:2301.11497 (2023)
54. Yu, F., Chen, Z., Li, M., Sanghi, A., Shayani, H., Mahdavi-Amiri, A., Zhang, H.: Capri-net: Learning compact cad shapes with adaptive primitive assembly. arXiv preprint arXiv:2104.05652 (2021)
55. Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. Advances in Neural Information Processing Systems (NeurIPS) (2022)