SweepNet: Unsupervised Learning Shape Abstraction via Neural Sweepers

Mingrui Zhao[1], Yizhi Wang[1], Fenggen Yu[1], Changqing Zou[2], and Ali Mahdavi-Amiri[1]

[1] Simon Fraser University
[2] Zhejiang University

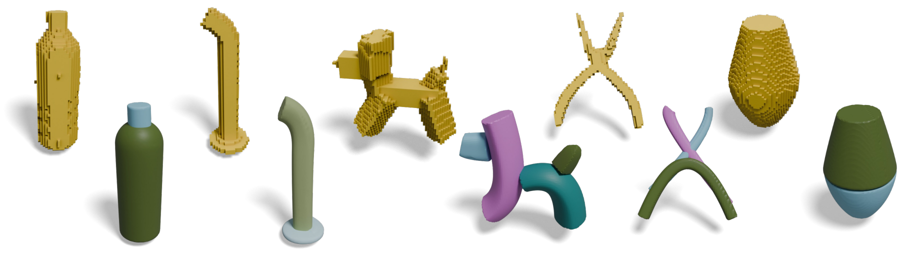

Fig. 1: Given a voxelized shape, SweepNet acquires abstraction without any supervision. With few sweep surfaces, the overall geometry of the objects is nicely captured.

Abstract. Shape abstraction is an important task for simplifying complex geometric structures while retaining essential features. Sweep surfaces, commonly found in human-made objects, aid in this process by effectively capturing and representing object geometry, thereby facilitating abstraction. In this paper, we introduce SweepNet , a novel approach to shape abstraction through sweep surfaces. We propose an effective parameterization for sweep surfaces, utilizing superellipses for profile representation and B-spline curves for the axis. This compact representation, requiring as few as 14 float numbers, facilitates intuitive and interactive editing while preserving shape details effectively. Additionally, by introducing a differentiable neural sweeper and an encoder-decoder architecture, we demonstrate the ability to predict sweep surface representations without supervision. We show the superiority of our model through several quantitative and qualitative experiments throughout the paper. Our code is available at `https://mingrui-zhao.github.io/SweepNet/`.

Keywords: Shape Abstraction · Primitive Fitting

1 Introduction

Sweep surfaces play an important role in computer graphics and computer vision, serving as fundamental constructs for modelling and analyzing complex shapes and structures. Sweep surfaces are extensively utilized for generating intricate geometric forms by sweeping a cross-sectional profile along a defined path. This enables the creation of diverse objects ranging from simple curves to intricate architectural designs, and more. On the other hand, shape abstraction is also an important problem in computer vision and graphics that involves representing complex geometric structures or objects in a simplified form while preserving essential characteristics for analysis or visualization purposes. Sweep surfaces can serve as a powerful tool for shape abstraction due to their ability to efficiently capture and represent the geometry of objects or structures due to their generality and ubiquity in the objects existing around us. By utilizing sweep surfaces, 3D shapes can be abstracted into more manageable representations, facilitating tasks such as shape recognition, and manipulation in various domains including computer graphics, or computer-aided design.

Current approaches to shape abstractions can be categorized by the type of constituent primitives. Popular choices include cuboid [54, 62, 73, 80], superquadrics [32, 33, 39, 51, 70, 76], parametric surfaces [53, 74], convex shapes [7, 12], neural parts [22, 25, 38], sketch-and-extrude [29, 45, 63], and a combination of simple primitives [19, 24, 27, 31, 44, 52]. However, each representation undergoes unique advantages and limitations. For instance, parametric primitives like cuboids and superquadrics offer ease of modification through parameter adjustments, facilitating interactivity. However, their simplicity often leads to less compact and expressive abstractions. Conversely, neural primitives showcase superior expressiveness by capturing complex shapes more accurately but suffer from reduced manipulability post-creation, which limits user control.

Shape abstraction via sweep surfaces can strike a balance between compactness and expresiveness (see Fig. 1). However, learning shape abstraction via sweep surfaces is challenging, primarily due to the limitations of accurate representation and the complexities involved in their parametrization. Existing methods of determining sweep surfaces are formulated as optimization problems [50] concerning sweep profile, sweep axis, and sweep motion, which makes it hard to integrate within a larger problem or a deep neural network. When integrated into computational methods, this introduces a nested optimization challenge, and within a learning framework, it results in a non-differentiable component.

In this paper, we first provide a simple parameterization for sweep surfaces that can be learned via a differentiable network. We employ superellipses for the profile representation due to its simplicity to learn and diversity in shape (see Fig. 3). For the axis, we use B-spline curves, in conjunction with basic polynomials to control sweep dynamics. Consequently, in our representation, a sweep surface can be represented with as few as 14 float numbers. Moreover, the complexity of these primitives can be easily scaled by expanding the parameter space. We then show how we can utilize this representation to learn shape abstractions for a given 3D object. Our approach involves an encoder-decoder architecture and a differentiable neural sweeper, enabling the model to predict the sweep surface representations *without any supervision.* By jointly optimizing representation faithfulness, sweeping rationality, and primitive parsimony, our model delivers high-quality abstractions of the target shape (Fig. 2 and Fig. 1). Therefore, our contributions are as follows:

– We provide the first deep learning model equipped with a differentiable sweeper specifically designed for shape abstraction through sweep surfaces.
– Our method introduces a new and compact parameterization of sweep surfaces, enabling intuitive and interactive editing.
– We demonstrate the advantages of our sweep surfaces over traditional parametric primitives in representing curvy-featuring objects, showcasing its superiority in achieving concise and expressive shape abstractions.

2 Related Work

Swept Volumes. Swept volume [1,5,46,50,57,66,78] refers to the total volume displaced by a moving object as it travels through a particular path or trajectory. The key challenges of constructing swept volumes involve not only constructing ruled surface patches for each edge and face but also trimming their mutual intersections to remove components not contributing to the final surface. Sèllan et al. [50] introduce spacetime numerical continuation for swept volume construction, offering enhanced generality and robustness with asymptotic complexity one order lower than prevailing industry standards. However, the construction process of swept volumes is typically non-differentiable, precluding its integration into our network. To address this challenge, we propose Neural Sweeper, a neural network designed to approximate implicit fields for swept volumes using profile and axis information as input. We leverage the methodology outlined in [50] for data preparation, wherein ground-truth mesh data of swept volumes is generated to calculate the occupancy field for training our neural network.
Neural Implicit Fields. OccNet [34], IM-Net [8], and DeepSDF [36] concurrently introduced the coordinate-based neural implicit representation. These early works only model global shape features, yielding over-smooth results which lack geometric details. The next wave in this direction has focused on conditioning implicit neural representations on local features stored in voxel [6,10,23,40,59], image grids [28,47,72] or surface points [4,17,21,68] to more effectively recover geometric or topological details and to scale to scene reconstruction, or at the patch level [15,61,67,77] to improve generalizability across object categories. We utilize POCO [4] as the backbone network of our neural sweeper, which takes as input the point cloud of the sweep surface to predict its implicit field.
Primitive Detection and Fitting. Traditional methods [3,13,30,35,43,49] for primitive detection involve RANSAC [14] and Hough Transform [20]. The work of Zou et al. [80] and Tulsiani et al [62] are among the earliest works that employ neural networks for primitive fitting, using cuboids as the only primitives. SPFN [27] and ParSeNet [53] consider unions of primitive patches to fit given 3D objects typically represented by point clouds. Constructive solid geometry (CSG) is a classical CAD representation which models a 3D shape as a recursive assembly of solid primitives using operations including union, intersection, and difference. Many recent works of primitive fitting are built on CSG trees, such as CSGNet [52], UCSG-NET [24], BSP-Net [7], CSG-Stump [44], CAPRI-Net [76], and D2CSG [75]. Inspired by the user-level construction sequence of

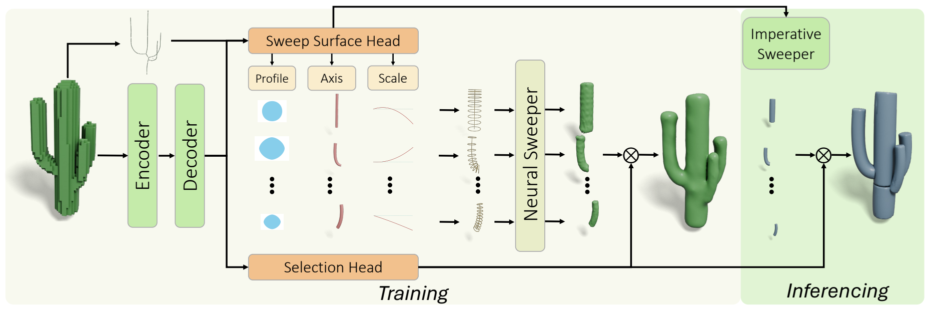

Fig. 2: Pipeline overview. The model processes voxel input to extract a skeletal prior and encodes the data with a voxel encoder. The sweep surface head predicts sweep surface parameters: 2D profiles, 3D sweeping axes, and profile scaling function coefficients, conditioned on the skeletal prior. Training involves generating point clouds for each sweep surface through a differentiable sampler, which the neural sweeper uses to estimate their occupancy. This data is then assembled to reconstruct the input shape to quantify loss. At inference time, the sweep surface parameters are directly processed by a non-differentiable, imperative sweeper to produce the resembled shape.

CAD models [69], Point2Cyl [63] and SECAD-Net [29] utilize sketch-and-extrude operations which enable the construction of 3D solid shapes from 2D sketches, which can ease the process of primitive fitting. We introduce a customized sketch-and-extrude process to create our sweep surfaces, with a superellipse as the 2D profile, a B-spline **curve** as the sweeping axis (direction) and a scaling function re-scaling the profile along the sweeping axis.
Shape Abstraction. Shape abstraction aims to fit 3D objects using *simple* and *compact* geometric primitives. Computational approaches [2, 9, 18, 26, 32, 33, 41, 55, 64, 70] directly optimize the parameters of primitives to fit a given shape. The primitives are typically superquadrics due to its extensive shape vocabulary including cuboids, ellipsoids, cylinders, octohedra, and many shapes in between. However, computational approaches typically rely on dense inputs (point clouds or SDFs) and require a closed-form equation for the implicit function of the primitives. Learning-based approaches [16,19,37,39,51,56,62,65,73] are versatile in dealing with different input sources, such as point clouds, voxel grids, or even RGB images. Since the implicit function of sweep surfaces has no closed-form equations, we learn a neural implicit field for them to approximate the function.

3 Method

This section presents an unsupervised model that parses 3D shapes using parametric sweep surfaces. We call our model SweepNet whose overall pipeline is outlined in Fig. 2. It encodes the input voxel shape using a 3D convolutional network to extract features, which are then enhanced through a three-layer MLP to produce a latent representation, z. This representation feeds into a dual MLP head: one predicting parameters for multiple sweep surfaces (sweep surface head)

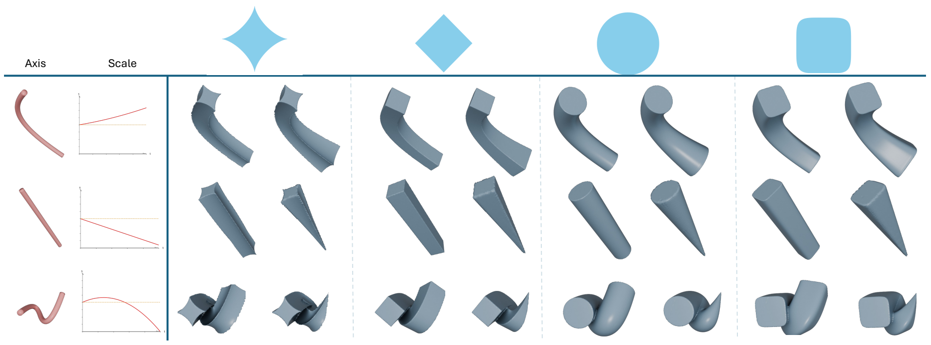

Fig. 3: Sweep surface primitives parameterized with different profiles, axis, and scaling functions. The 2D profiles (superellipses) are shown in blue. Constant and dynamic scales of sweep surfaces are shown in alternating columns with respect to each profile and axis pair.

and another selecting a subset of these primitives (selection head) for parsimonious shape assembly. During this process, the prediction of the sweeping axes is guided by the medial axis [58] of the input voxel shape. For each sweep surface, we select a series of points along its sweeping axis as well as a number of profile slices for examination. These profile slices are created by projecting the 2D profile onto the 3D sweeping curve, utilizing the curve's coordinate frame for transformation. The resulting point cloud is a compilation of points from both the sweeping axis and these transformed loops. This gathered data is then processed by the neural sweeper to predict the occupancy field of the sweep surface. The final assembled shape is presented by the union of the selected sweep surface occupancies. At inference time, the predicted primitive parameters are used directly to produce the final sweep surfaces using standard but non-differentiable sweepers, bypassing the neural sweeper, for efficient primitive production, and the parsed shape is compactly described by their parameters in the size of only tens of floats.

3.1 Sweep Surface Primitive

A sweep surface is defined by a 2D profile, a 3D sweeping axis, and a function f controlling the profile's scale along the axis. Here, we target for a parameterizable, compact and expressive representation for all three components. A 2D profile is defined as a finite closed loop with no self-intersections. Existing works attempted implicit fields [29], neural sketches [63] and rational Bézier polygons [45] to represent such profiles. Implicit fields can well represent complex profiles, however, they are carried through a neural network and hence offer limited editability post-creation. The rational Bezier polygon is parameterizable and offers good expressiveness. However, it needs additional constraints in formulation to maintain the non-self-intersection property, requires critical coefficients, and is sometimes impossible, to represent certain regular shapes such as rectangles. Alternatively, we choose superellipse to parameterize the 2D profile, formulated as

$$\begin{cases} x(\theta) = a \cdot |\cos(\theta)|^{\frac{2}{d}} \cdot \operatorname{sgn}(\cos(\theta)), \\ y(\theta) = b \cdot |\sin(\theta)|^{\frac{2}{d}} \cdot \operatorname{sgn}(\sin(\theta)), \end{cases} \tag{1}$$

where $\theta \in [0, 2\pi]$ is the polar angle, $x, y \in \mathbb{R}$ are the Cartesian superellipse contour coordinates, $a, b \in \mathbb{R}^+$ represents the major and minor axis and $d \in \mathbb{R}^+$ stands for the curvature degree. A superellipse is parameter-compact with as few as 3 parameters, naturally preserves self-intersection, and offers a flexible representation from rectangular to star shapes. It offer a straightforward way to model essential shapes such as squares and circles, striking a balance between parameter simplicity and representation versatility

For the 3D sweeping axis, our methodology employs a third-order B-spline characterized by n control points $\{c_1, \cdots, c_n\} \in \mathbb{R}^3$, ensuring a flexible yet precise control over the shape's curvature. The spline is parametrized over a clamped knot vector, guaranteeing that the B-spline's trajectory starts at c_1 and concludes at c_n. The spline curvature is modulated by the intermediate control points positions. In scenarios where these control points align collinearly, the sweep axis simplifies to a straight line, accommodating the fundamental sketch-and-extrude cases.

To address the inherent rotation ambiguity of the profile during sweeping, we adopt the parallel transport frames to regularise the sweeping motion. Fig. 4 (a), (b) show two sweep surfaces produced with the same profile and axis but with different sweeping motions. The convention of the sweep motion is aligning the profile normal (z-axis) with the sweeping trajectory tangent, which still leaves the $x - y$ axis of the profile indeterminate. The parallel transport coordinate frame establishes consistent local coordinate frames along the B-spline curve, eliminating the ambiguity in profile orientations. This method is particularly effective in maintaining consistent and deterministic sweeping motion, while being robust against extreme curvature scenarios, such as inflection points. As a result, the translation and rotation of the profile are uniquely derived along the sweeping motion, enhancing the model's precision and reliability.

The last component, scaling function f, dynamically adjusts the profile scale along the sweep. Defined as $f(t) : [0, 1] \to \mathbb{R}^+$ for a sweeping axis $s(t) : [0, 1] \to \mathbb{R}^3$, it ensures the profile is appropriately scaled at each point $s(t)$. To achieve a smooth and continuous sweep, f needs to be strictly C^0 continuous. We choose degree k polynomials with a fixed constant term to formulate a scaling function, offering both effective scaling behaviour and a compact parametrization. Collectively, a sweep surface primitive is uniquely defined by:

$$S = [c_1, \cdots, c_n, a, b, d, f_1, \cdots, f_k] \in \mathbb{R}^{3n+k+3} \tag{2}$$

Figure 3 showcases sweep surfaces with various profiles, sweeping axes, and in different combinations with the scaling function.

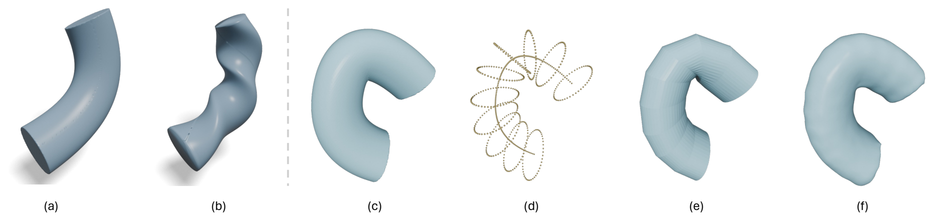

Fig. 4: Sweep surfaces visualised under various conditions. (a): Sweep surface produced with parallel-transportation frame regularization. (b): Sweep surface produced with free profile rotations. (c): Reference sweep surface produced by off-the-shelf sweeper. (d) Point cloud sampled from sweep surfaces for neural sweeper input. (e) Sweep surface produced by naively interpolating profile stamps, notice the sharp crease and aliasing effect at high curvature regions. (f) Sweep surface produced by neural sweeper with the input point cloud in (d).

3.2 Neural Sweeper

The integration of sweep surface primitives into a learning framework is complicated by the lack of a differentiable method to generate these primitives from their defining parameters. Traditional approaches for creating sweep surfaces involve densely sampling profile frames along the sweeping axis. This sampling is followed by one of two methods: connecting adjacent profile points to construct an explicit swept volume mesh [11] or employing numerical continuation to compute an implicit swept volume field as the profile traverses the sweeping axis [50]. Unfortunately, both methods present integration challenges within a learning context due to their non-differentiable nature. Moreover, a direct analytical approach, involving dense sampling and occupancy interpolation between profile frames, incurs significant computational costs and is susceptible to aliasing effects, particularly with sharply curved profiles. An example is provided in Fig. 4 (e), where the sweep surface produced from interpolation shows visible coarse granularity and sharp creases at high curvature regions.

In response to these challenges, we introduce the concept of a neural sweeper, a differentiable surrogate for sweep surface generation. This approach begins with the use of a differentiable sampler to collect sweep surface key points. The key points are collected by sampling points along the sweeping axis, and 3D profile slices by sparsely transforming the 2D profile into 3D space based on curve coordinate frames (Fig. 4 (d)). These sampled points are then processed by the neural sweeper to compute the corresponding implicit field. For model training, we generated a dataset comprising sweep surface samples with varied parameters, and applied the existing point-cloud-to-implicit-field model, POCO [4], as the backbone model. Once the model is trained, we freeze the neural sweeper and plug it into SweepNet to facilitate subsequent outer scope training sessions. The selected architecture of the neural sweeper must support smooth gradient flow across interfaces with other components. We observed that the use of a discretized feature grid and stochastic sampling can impede gradient flow, potentially obstructing the training process of the larger framework. However, the POCO model is particularly advantageous in this context, capable of capturing detailed implicit shape features while maintaining compatibility with back-propagation techniques, thereby serving as an effective submodule within our proposed methodology.

3.3 Training and Inference

SweepNet implements a two-phase training strategy: initially, the neural sweeper is trained, followed by the comprehensive training of SweepNet itself. The training of the neural sweeper utilizes Binary Cross-Entropy loss to evaluate the accuracy of the predicted occupancy fields for sweep surfaces. Training SweepNet model involves a blend of reconstruction loss, axis loss, overlap loss, and parsimony loss to optimize shape parsing.

With K sweep surfaces predicted, the neural sweeper generates occupancy fields $O_1, \cdots, O_K$. From these, SweepNet selects a subset of primitives $p_1, \cdots, p_q$ (where $q \le K$) to construct the final shape. Testing points T spread throughout the 3D space are used to calculate the reconstruction loss, which is the mean squared error between the ground truth occupancy field and the assembled occupancy field. The latter is derived using the Boltzmann operator with a sharpness parameter $\alpha \in \mathbb{R}$:

$$\mathcal{L}_{recon} = \mathbb{E}_{t \sim T} \left[\left\| O_{GT}(t) - \frac{\sum_{i=1}^{q} O_i(t) e^{\alpha O_i(t)}}{\sum_{i=1}^{q} e^{\alpha O_i(t)}} \right\|_2^2 \right]. \tag{3}$$

Furthermore, to ensure a parsimonious representation, SweepNet minimizes the use of primitives through both overlap loss and parsimony loss. The overlap loss penalizes excessive overlapping among sweep surface primitives beyond a threshold β:

$$\mathcal{L}_{ol} = \mathbb{E}_{t \sim T} \left[\min(\sum_{i=1}^{q} O_i(t) - \beta, 0) \right]. \tag{4}$$

The parsimony loss, encouraging minimal primitive usage, is represented as a sublinear function of the count of selected primitives:

$$\mathcal{L}_{pars} = \sqrt{q}. \tag{5}$$

Axis loss is introduced to guide the prediction of sweeping axes, aligning them closely with the object's medial axis. This is crucial as unsupervised learning of sweep axes is inherently ambiguous due to the multitude of possible profile-axis combinations that can generate the same object. The learning is regularized by ensuring the predicted sweeping axes encompass the object's medial axis, quantified by the chamfer distance between the medial axis points $M = m_i$ and the points sampled from selected sweeping axis $S = \cup_{i=1}^{q} \{s_{ij}\}$:

$$\mathcal{L}_{axis} = \mathbb{E}_{m \sim M} \left[\min_{s \in S} dist(m, s) \right]. \tag{6}$$

The overall SweepNet loss function is defined as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{recon} + \lambda_2 \mathcal{L}_{ol} + \lambda_3 \mathcal{L}_{pars} + \lambda_4 \mathcal{L}_{axis}. \tag{7}$$

Before the training starts, the sweep surface primitives are initialized regarding the medial axis for a warm start. At inference time, we use off-the-shelf sweepers [50] to create explicit sweep surface primitives from the predicted parameters, bypassing the neural sweeper to improve speed and accuracy. Empirically, we set $\lambda_1 = 12$, $\alpha = 40$, $\lambda_2 = 6$, $\beta = 0.8 \cdot K$, $\lambda_3 = 0.3K/8$ and $\lambda_4 = 5$. More details about the hyper-parameter setting and training practice can be found in the supplemental material.

4 Results

In this section, we begin by offering comprehensive insights into our datasets and implementation methodologies. Subsequently, we present both quantitative metrics and qualitative observations of our method compared to alternative approaches. We also provide ablation studies to justify our design choices. More results and ablations will be provided in the supplementary material.

4.1 Dataset and Implementation Details

We conduct experiments over two datasets, a custom *GC-Object* dataset containing 50 models sourced from prior works [48,79] and internet; and quadrupeds dataset [62,71] with 124 animal shapes. The data are preprocessed following the scheme of CAPRI-NET [76].

We showcase the parametric lightness and sweep-versatility properties of sweep surfaces. To this end, we compare SweepNet with several baseline models: two primitive-fitting shape abstraction methods using superquadrics (SQ) [39] and cuboids (Cuboid) [73], one network using sketch-and-extrude primitives with neural profiles (SECAD-Net) [29], one network using sketch-and-extrude primitives with CSG operations (ExtrudeNet) [45] and one network using geons with CSG operations (UCSG) [24]. In addition, we provide insight on SweepNet with point cloud input modality by switching the encoder to DGCNN [42] module, denoted as Ours_{pcd}, showcasing the flexibility of our pipeline.

Our intention with this comparison is to demonstrate that, within a comparable training period, SweepNet produces superior results with minimal training iterations. Sweep surfaces exhibit greater expressiveness and versatility when dealing with curvy objects compared to conventional parametric and sketch-and-extrude primitives. This highlights the necessity of introducing sweep surfaces as a new primitive for shape abstraction

Since our model works on a per-shape basis, we empirically adapt the training scheme for the baseline models to accommodate single-shape fitting. We prioritize using the default setups for each baseline model. If they do not converge well within this setup, we increase the training iterations and/or enhance supervision signals, capped at a maximum of 10 minutes per shape training cost on an NVIDIA RTX4090 GPU.

For SQ, we train the model on each input shape for 4,000 iterations. For Cuboid, we train the model on each input shape for 10,000 iterations. For SECAD-Net, we pretrain the model on the entire dataset for 1,000 epochs, followed by fine-tuning on each model for another 2,000 iterations before inference. ExtrudeNet and UCSG require longer training epochs to learn CSG operations. For ExtrudeNet, we replace the voxel input with a point cloud of 32,764 points and provide 100,000 occupancy points for supervision (3.05 $\times$ our input). ExtrudeNet is trained for 60,000 iterations, and UCSG is trained for 40,000 iterations. For SweepNet, we train the model on each input shape for 2,000 iterations without any pre-training. All models are trained with a maximum of eight primitives.

4.2 Quantitative Comparisons

We present quantitative measurements obtained for Chamfer-Distance (CD), Volumetric Intersection over Union (IoU), and F-score with an accuracy threshold of 0.05 (F1) [60].

The detailed quantitative results are presented in Table 1 for the GC-Object dataset and in Table 2 for quadrupeds. Across all three metrics provided for the GC-Object dataset, our method outperforms others. In the quadrupeds dataset, our method demonstrates superior performance in all metrics compared to other methods, except for IoU against SQ [39], where our method falls slightly short.

Table 1: Quantitative evaluations on GC-Object dataset. Our method outperforms other alternatives in all the metrics included.

	EN	UCSG	SECAD	Cuboid	SQ	Ours	Ours$_{\text{pcd}}$
IoU ↑	0.420	0.554	0.584	0.343	0.597	0.608	0.620
CD ↓	0.0481	0.0170	0.0179	0.0280	0.0182	0.0168	0.0168
F1 ↑	0.652	0.947	0.967	0.871	0.977	0.984	0.985

Table 2: Quantitative evaluations on quadrupeds. Our method with voxel input slightly falls short of SQ [39] while leading with point cloud input.

	EN	UCSG	SECAD	Cuboid	SQ	Ours	Ours$_{\text{pcd}}$
IoU ↑	0.197	0.436	0.270	0.259	0.512	0.482	0.562
CD ↓	0.0668	0.0177	0.0224	0.0313	0.0179	0.0176	0.0168
F1 ↑	0.458	0.952	0.932	0.840	0.964	0.967	0.989

4.3 Qualitative Comparisons

The qualitative results are demonstrated in Fig. 10, it can be seen that sweep surfaces, with appropriate scaling, have superior expressiveness. Sweep elements like curvy-linear limbs can be compactly represented by sweep surfaces using a single primitive, whereas other baseline primitives require multiple components for approximation. By leveraging the sweeping axis from straight lines to curves, sweep surfaces faithfully represent tubular parts such as ant legs which are challenging for sketch-and-extrude. Additionally, the scaling function enhances the versatility of sweep surfaces, effectively capturing shapes such as the gradually thinning gecko tail and the cone shape in the icecream.

4.4 Ablation Studies

We conducted an extensive ablation study on our total loss designs, deactivating each loss component one at a time. In each experimental setting, we trained our model for a fixed iteration of 1,000 and simultaneously assessed the convergence speed. The qualitative outcomes are depicted in Fig. 5.

As illustrated in Fig. 5, employing the full loss incorporating all four components yields results that closely mirror the input. When $\mathcal{L}_{pars}$ is disabled, there is a noticeable increase in the utilization of primitives to compensate for fine-scale details. Similarly, omitting the overlap loss $\mathcal{L}_{ol}$ leads to the emergence of undesired overlaps, particularly evident in the body of the dog. Furthermore, excluding the axis loss $\mathcal{L}_{axis}$ results in a loss of fidelity in preserving the curvy shape of the dog, yielding a more cumbersome appearance. These observations underscore the significance of each component within our loss framework. Additional results can be found in the supplementary material.

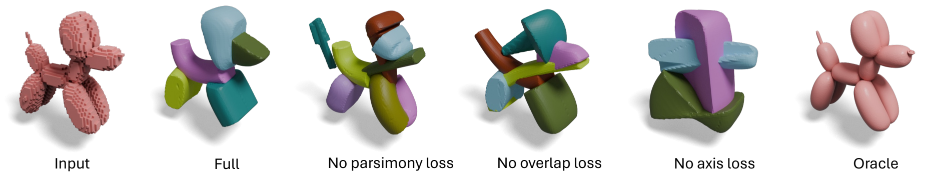

Fig. 5: Shape abstraction results with various loss settings. Our model performs the best with all loss functions equipped.

We additionally assess the impact of the parametric complexity of sweep surfaces by increasing the number of control points from 3 to 4. A qualitative example is demonstrated in Fig. 6. With both options, the model can make a reasonable abstraction of the input shape. Although our standard configuration utilizes 3 control points for our B-spline axis, our method also performs effectively with 4 control points. This indicates the relative robustness of our method against variations in the number of axis control points.

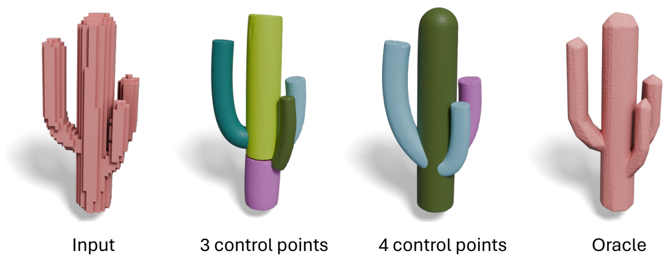

Fig. 6: Ablation study showing how the number of control points affects the abstraction results. While our default is to have 3 control points for our B-spline axis, our method performs well with 4 control points too.

Lastly, we conduct a sensitivity test on the medial axis. The medial axis is a crucial component in SweepNet, leading to a faster and more rational fitting of sweep surfaces. Despite the reliance of SweepNet on this skeletal prior, our method exhibits a certain degree of robustness against noisy medial axes. We showcase two examples in Figure 7. In the first example, we inject Gaussian noise with a standard deviation of 0.01 to the extracted gecko medial axis. In the second example, the extracted medial axis of the octopus is incomplete, missing the head. In both cases, SweepNet can compensate for the faulty part and produce reasonable abstracted results.

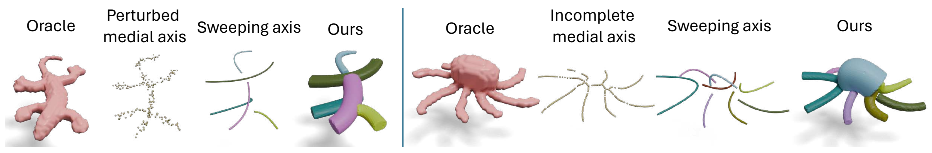

Fig. 7: Shape abstraction results with noisy or incomplete medial axis guidance.

4.5 Editablity

In this section, we illustrate the flexibility of parametric sweep surfaces through post-creation editing. We present a case study in Fig. 8, where the faucet valve is rotated by applying an affine transformation to the sweeping axis control point coordinates, demanding a change to only 9 float numbers. We provide additional edit examples in the supplementary material.

5 Conclusion, Limitations, and Future Work

Our method faces some limitations and future directions can be explored. The current model falls short in representing high-porosity or overly thin objects with

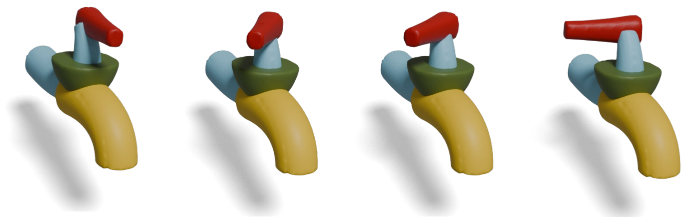

Fig. 8: Primitive editing of spinning faucet valve by altering sweep surface parameters.

solid sweep surfaces. Moreover, shape abstraction tends to struggle when dealing with complex models containing numerous intricate details (refer to Fig. 9). Our method performs optimally when the provided model includes sweep elements. If the model lacks such elements, our method may not achieve the most favourable outcome. Hence, an intriguing area for exploration lies in integrating neural sweepers with other types of primitives within more complex systems (e.g., CSG techniques) to capture geometric intricacies while maintaining compactness. Also, currently SweepNet fits to each model individually, which can encounter local optimums at different initialization, future research can be done to extend this work for a generalizable shape abstraction model.

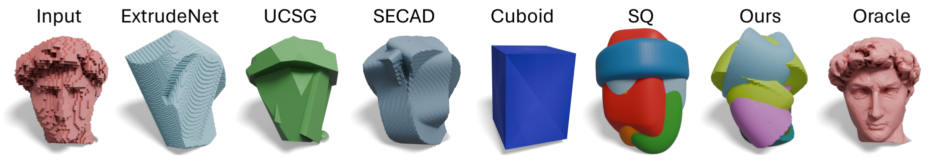

Fig. 9: Failure cases of shape abstraction methods.

In this paper, we presented SweepNet , a method designed for shape abstraction through the utilization of sweep surfaces. Our approach introduces a novel and compact parameterization that facilitates intuitive editing and effectively retains shape details. The integration of neural sweepers introduces a new way to incorporate challenging primitives into shape abstraction tasks. Neural sweepers can be seamlessly plugged and played in other deep learning networks for sweep surface production or tailored to tackle other complex geometric primitives, providing a versatile tool for advancing shape abstraction techniques. Collectively, our model showcases its ability to accurately predict shape abstractions via sweep surfaces without the need for supervision. We have also demonstrated the superiority of our approach over conventional methods in shape abstraction targeting on curvy-feature objects. In conclusion, SweepNet is a step further in 3D shape abstraction, combining the strengths of sweep surfaces with efficient parameterization and dynamic scaling. While there are limitations to address, the potential for future enhancements is vast.

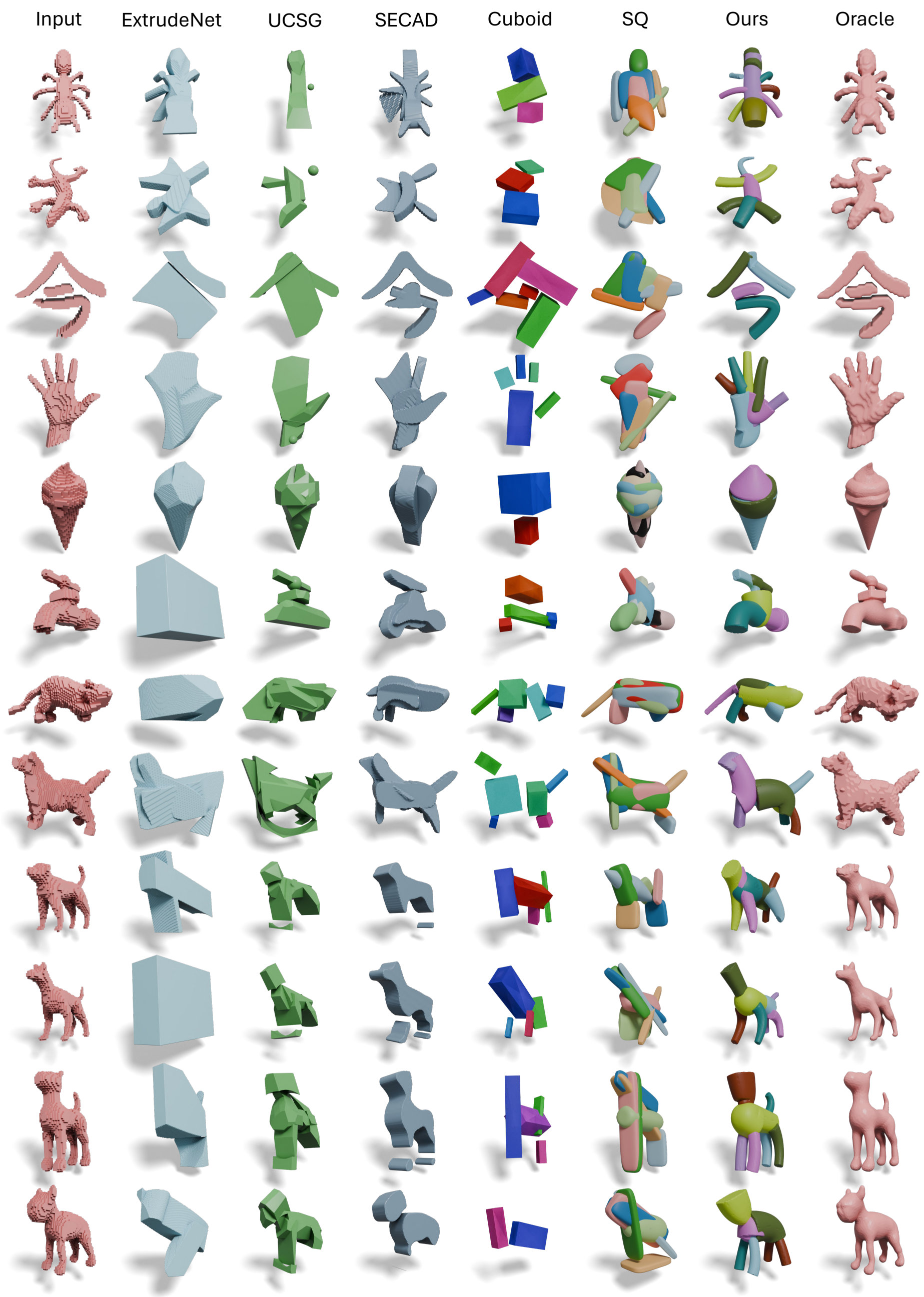

Fig. 10: Qualitative comparison among ExtrudeNet [45], UCSG [24], SECAD-Net [29], Cuboid [73], SQ [39] and our method. Models abstracted by our method requires less primitives while better representing curvy geometric features.

Acknowledgement

We thank all anonymous reviewers for their invaluable feedback and constructive comments. We thank Cody Reading for the insightful discussions and Ruiqi Wang for the help with figure preparations.

References

1. Abdel-Malek, K., Yang, J., Blackmore, D., Joy, K.: Swept volumes: fundation, perspectives, and applications. International Journal of Shape Modeling **12**(01), 87–127 (2006)
2. Barr, A.H.: Superquadrics and angle-preserving transformations. IEEE Computer graphics and Applications **1**(1), 11–23 (1981)
3. Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A.: The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. 3D Research **2**(2), 1–13 (2011)
4. Boulch, A., Marlet, R.: Poco: Point convolution for surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6302–6314 (2022)
5. Campen, M., Kobbelt, L.: Polygonal boundary evaluation of minkowski sums and swept volumes. In: Computer Graphics Forum. vol. 29, pp. 1613–1622. Wiley Online Library (2010)
6. Chabra, R., Lenssen, J.E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., Newcombe, R.: Deep local shapes: Learning local sdf priors for detailed 3D reconstruction. In: ECCV (2020)
7. 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)
8. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: CVPR (2019)
9. Chevalier, L., Jaillet, F., Baskurt, A.: Segmentation and superquadric modeling of 3d objects (2003)
10. Chibane, J., Alldieck, T., Pons-Moll, G.: Implicit functions in feature space for 3D shape reconstruction and completion. In: CVPR (2020)
11. Dawson-Haggerty et al.: trimesh, `https://trimesh.org/`
12. 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)
13. Drost, B., Ilic, S.: Local hough transform for 3d primitive detection. In: 2015 International Conference on 3D Vision. pp. 398–406. IEEE (2015)
14. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
15. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Local deep implicit functions for 3D shape. In: CVPR (2020)
16. Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W.T., Funkhouser, T.: Learning shape templates with structured implicit functions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7154–7164 (2019)

17. Giebenhain, S., Goldlücke, B.: Air-nets: An attention-based framework for locally conditioned implicit representations. In: 2021 International Conference on 3D Vision (3DV). pp. 1054–1064. IEEE (2021)
18. Gross, A.D., Boult, T.E.: Error of fit measures for recovering parametric solids. In: 1988 Second International Conference on Computer Vision. pp. 690–691. IEEE Computer Society (1988)
19. 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)
20. Hough, P.V.: Machine analysis of bubble chamber pictures. In: International Conference on High Energy Accelerators and Instrumentation, CERN, 1959. pp. 554–556 (1959)
21. Huang, J., Gojcic, Z., Atzmon, M., Litany, O., Fidler, S., Williams, F.: Neural kernel surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4369–4379 (2023)
22. Huang, X., Zhang, Y., Chen, K., Li, T., Zhang, W., Ni, B.: Learning shape primitives via implicit convexity regularization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3642–3651 (2023)
23. Jiang, C.M., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T.: Local implicit grid representations for 3D scenes. In: CVPR (2020)
24. Kania, K., Zieba, M., Kajdanowicz, T.: Ucsg-net-unsupervised discovering of constructive solid geometry tree. Advances in Neural Information Processing Systems **33**, 8776–8786 (2020)
25. Kawana, Y., Mukuta, Y., Harada, T.: Neural star domain as primitive representation. Advances in Neural Information Processing Systems **33**, 7875–7886 (2020)
26. Leonardis, A., Jaklic, A., Solina, F.: Superquadrics for segmenting and modeling range data. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**(11), 1289–1295 (1997)
27. Li, L., Sung, M., Dubrovina, A., Yi, L., Guibas, L.J.: Supervised fitting of geometric primitives to 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2652–2660 (2019)
28. Li, M., Zhang, H.: D^{2}IM-Net: Learning detail disentangled implicit fields from single images. In: CVPR (2021)
29. 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)
30. Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J.: Globfit: Consistently fitting primitives by discovering global relations. In: ACM SIGGRAPH 2011 papers, pp. 1–12 (2011)
31. Li, Y., Liu, S., Yang, X., Guo, J., Guo, J., Guo, Y.: Surface and edge detection for primitive fitting of point clouds. In: ACM SIGGRAPH 2023 conference proceedings. pp. 1–10 (2023)
32. Liu, W., Wu, Y., Ruan, S., Chirikjian, G.S.: Robust and accurate superquadric recovery: A probabilistic approach. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2676–2685 (2022)
33. Liu, W., Wu, Y., Ruan, S., Chirikjian, G.S.: Marching-primitives: Shape abstraction from signed distance function. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8771–8780 (2023)
34. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR. pp. 4460–4470 (2019)

35. Oesau, S., Lafarge, F., Alliez, P.: Planar shape detection and regularization in tandem. In: Computer Graphics Forum. vol. 35, pp. 203–215. Wiley Online Library (2016)
36. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning continuous signed distance functions for shape representation. In: CVPR. pp. 165–174 (2019)
37. Paschalidou, D., Gool, L.V., Geiger, A.: Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1060–1070 (2020)
38. Paschalidou, D., Katharopoulos, A., Geiger, A., Fidler, S.: Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3204–3215 (2021)
39. 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)
40. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: ECCV (2020)
41. Pentland, A.P.: Perceptual organization and the representation of natural form. In: Readings in Computer Vision, pp. 680–699. Elsevier (1987)
42. Phan, A.V., Le Nguyen, M., Nguyen, Y.L.H., Bui, L.T.: Dgcnn: A convolutional neural network over large-scale labeled graphs. Neural Networks **108**, 533–543 (2018)
43. Rabbani, T., Van Den Heuvel, F.: Efficient hough transform for automatic detection of cylinders in point clouds. Isprs Wg Iii/3, Iii/4 **3**, 60–65 (2005)
44. 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. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12478–12487 (2021)
45. 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)
46. Rossignac, J., Kim, J.J., Song, S., Suh, K., Joung, C.: Boundary of the volume swept by a free-form solid in screw motion. Computer-Aided Design **39**(9), 745–755 (2007)
47. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: ICCV (2019)
48. Sawdayee, H., Vaxman, A., Bermano, A.H.: Orex: Object reconstruction from planar cross-sections using neural fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20854–20862 (2023)
49. Schnabel, R., Wahl, R., Klein, R.: Efficient ransac for point-cloud shape detection. In: Computer graphics forum. vol. 26, pp. 214–226. Wiley Online Library (2007)
50. Sellán, S., Aigerman, N., Jacobson, A.: Swept volumes via spacetime numerical continuation. ACM Transactions on Graphics (TOG) **40**(4), 1–11 (2021)
51. Sharma, G., Dash, B., RoyChowdhury, A., Gadelha, M., Loizou, M., Cao, L., Wang, R., Learned-Miller, E., Maji, S., Kalogerakis, E.: Prifit: learning to fit primitives improves few shot point cloud segmentation. In: Computer Graphics Forum. vol. 41, pp. 39–50. Wiley Online Library (2022)

52. Sharma, G., Goyal, R., Liu, D., Kalogerakis, E., Maji, S.: Csgnet: Neural shape parser for constructive solid geometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5515–5523 (2018)
53. Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., Měch, R.: Parsenet: A parametric surface fitting network for 3d point clouds. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16. pp. 261–276. Springer (2020)
54. Smirnov, D., Fisher, M., Kim, V.G., Zhang, R., Solomon, J.: Deep parametric shape predictions using distance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 561–570 (2020)
55. Solina, F., Bajcsy, R.: Recovery of parametric models from range images: The case for superquadrics with global deformations. IEEE transactions on pattern analysis and machine intelligence **12**(2), 131–147 (1990)
56. Sun, C.Y., Zou, Q.F., Tong, X., Liu, Y.: Learning adaptive hierarchical cuboid abstractions of 3d shape collections. ACM Transactions on Graphics (TOG) **38**(6), 1–13 (2019)
57. Sungurtekin, U., Voelcker, H.: Graphical simulation & automatic verification of nc machining programs. In: Proceedings. 1986 IEEE International Conference on Robotics and Automation. vol. 3, pp. 156–165. IEEE (1986)
58. Tagliasacchi, A., Alhashim, I., Olson, M., Zhang, H.: Mean curvature skeletons. In: Computer Graphics Forum. vol. 31, pp. 1735–1744. Wiley Online Library (2012)
59. Tang, J., Lei, J., Xu, D., Ma, F., Jia, K., Zhang, L.: Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6504–6513 (2021)
60. Tatarchenko, M., Richter, S.R., Ranftl, R., Li, Z., Koltun, V., Brox, T.: What do single-view 3d reconstruction networks learn? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3405–3414 (2019)
61. Tretschk, E., Tewari, A., Golyanik, V., Zollhoefer, M., Stoll, C., Theobalt, C.: PatchNets: Patch-based generalizable deep implicit 3D shape representations. In: ECCV (2020)
62. 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)
63. Uy, M.A., Chang, Y.Y., Sung, M., Goel, P., Lambourne, J.G., Birdal, T., Guibas, L.J.: Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11850–11860 (2022)
64. Vaskevicius, N., Birk, A.: Revisiting superquadric fitting: A numerically stable formulation. IEEE transactions on pattern analysis and machine intelligence **41**(1), 220–233 (2017)
65. Vora, A., Gadi Patil, A., Zhang, H.: Divinet: 3d reconstruction from disparate views using neural template regularization. Advances in Neural Information Processing Systems **36** (2024)
66. Wang, W., Wang, K.: Geometric modeling for swept volume of moving solids. IEEE Computer graphics and Applications **6**(12), 8–17 (1986)
67. Wang, Y., Huang, Z., Shamir, A., Huang, H., Zhang, H., Hu, R.: Aro-net: Learning implicit fields from anchored radial observations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3572–3581 (2023)
68. Williams, F., Gojcic, Z., Khamis, S., Zorin, D., Bruna, J., Fidler, S., Litany, O.: Neural fields as learnable kernels for 3d reconstruction. In: Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18500–18510 (2022)

69. Wu, R., Xiao, C., Zheng, C.: Deepcad: A deep generative network for computer-aided design models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6772–6782 (2021)
70. Wu, Y., Liu, W., Ruan, S., Chirikjian, G.S.: Primitive-based shape abstraction via nonparametric bayesian inference. In: European Conference on Computer Vision. pp. 479–495. Springer (2022)
71. Xu, J., Zhang, Y., Peng, J., Ma, W., Jesslen, A., Ji, P., Hu, Q., Zhang, J., Liu, Q., Wang, J., et al.: Animal3d: A comprehensive dataset of 3d animal pose and shape. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9099–9109 (2023)
72. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In: NeurIPS. pp. 492–502 (2019)
73. Yang, K., Chen, X.: Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. ACM Transactions on Graphics (TOG) **40**(4), 1–11 (2021)
74. Yang, L., Liang, Y., Li, X., Zhang, C., Lin, G., Sheffer, A., Schaefer, S., Keyser, J., Wang, W.: Neural parametric surfaces for shape modeling. arXiv preprint arXiv:2309.09911 (2023)
75. Yu, F., Chen, Q., Tanveer, M., Mahdavi Amiri, A., Zhang, H.: D2csg: Unsupervised learning of compact csg trees with dual complements and dropouts. Advances in Neural Information Processing Systems **36** (2024)
76. 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. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11768–11778 (2022)
77. Zhang, B., Nießner, M., Wonka, P.: 3dilg: Irregular latent grids for 3d generative modeling. arXiv preprint arXiv:2205.13914 (2022)
78. Zhang, X., Kim, Y.J., Manocha, D.: Reliable sweeps. In: 2009 SIAM/ACM joint conference on geometric and physical modeling. pp. 373–378 (2009)
79. Zhou, Y., Yin, K., Huang, H., Zhang, H., Gong, M., Cohen-Or, D.: Generalized cylinder decomposition. ACM Trans. Graph. **34**(6), 171–1 (2015)
80. Zou, C., Yumer, E., Yang, J., Ceylan, D., Hoiem, D.: 3d-prnn: Generating shape primitives with recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 900–909 (2017)