SfmCAD: Unsupervised CAD Reconstruction by Learning Sketch-based Feature Modeling Operations

Pu Li, Jianwei Guo, Huibin Li, Bedrich Benes, Dong-Ming Yan

Abstract

This paper introduces SfmCAD, a novel unsupervised network that reconstructs 3D shapes by learning the Sketch-based Feature Modeling operations commonly used in modern CAD workflows. Given a 3D shape represented as voxels, SfmCAD learns a neural-typed sketch+path parameterized representation, including 2D sketches of feature primitives and their 3D sweeping paths without supervision, for inferring feature-based CAD programs. SfmCAD employs 2D sketches for local detail representation and 3D paths to capture the overall structure, achieving a clear separation between shape details and structure. This conversion into parametric forms enables users to seamlessly adjust the shape’s geometric and structural features, thus enhancing interpretability and user control. We demonstrate the effectiveness of our method by applying SfmCAD to many different types of objects, such as CAD parts, ShapeNet objects, and tree shapes. Extensive comparisons show that SfmCAD produces compact and faithful 3D reconstructions with superior quality compared to alternatives. The code is released at https://github.com/BunnySoCrazy/SfmCAD.

1. Introduction

Reconstructing shapes into CAD representations is a fundamental problem in computer vision and graphics with numerous commercial applications in industrial design and manufacturing [39]. Recent years have seen an increased interest in leveraging deep learning techniques to learn geometric representations of 3D shapes, including implicit fields [2, 25, 28], geometric primitives [19, 22, 24, 37], boundary representation (B-Rep) [9, 17], or Constructive Solid Geometry (CSG) [3, 36, 49], each with its unique characteristics and applications, contributing to the rich tapestry of research and development in this field.

However, current CAD reconstruction methods are still lacking detail reconstruction accuracy, compactness, and interpretability of the obtained models. For example, a typical line of research is dedicated to reconstruction-oriented tasks, i.e., representing 3D shapes as parametric primitives, implicit fields, or B-rep. These methods deliver a high level of detail and accuracy by capturing intricate details and complex geometries with remarkable precision. However, constructing CAD models by assembling instances of primitives or by identifying faces, edges, and vertices is a tedious process [35]. Besides, the results often include complex and unintuitive operations that make the model difficult to inter-
prett and edit. This lack of control poses significant challenges in scenarios where user interaction and shape manipulation are required.

A straightforward approach to CAD reconstruction is to learn CSGi representation [14, 30, 36, 49] that implicitly recovers the CAD design history. A CSG 3D shape is expressed as a tree with leaves consisting of volumetric primitives (e.g., cubes, spheres, and cylinders) that are combined using Boolean operations in the CSG inner nodes. Despite the sought-after attribute of compact shape representation, these methods often fall short regarding reconstruction accuracy by using only a set of simple typed primitives.

We introduce SfmCAD, a novel neural model for reconstructing 3D shapes into their CAD representation. Compared to related work, SfmCAD improves the reconstruction accuracy and the level of model editability by decoupling 3D structures and local geometric details of the shape. SfmCAD is inspired by feature-based modeling, a predominant method for creating 3D models in modern CAD systems, which sequentially add features (e.g., holes, slots, bosses) that represent common manufacturing operations [6, 41, 42]. Specifically, we employ unsupervised learning to retrieve a neural typed sketch+path representation of 3D shapes. Taking 3D voxel data as input, SfmCAD learns implicit fields that embody the sketch profile and generates control points for a Bézier curve, constituting the sweeping path for the component’s geometry. The basic extrusion operation has been learned in a supervised [18, 38] or unsupervised [23, 31] manner. However, our sketch+path representation supports not only extrude but also sweep, loft, and revolve operations, thus enhancing the shape representation capabilities, as demonstrated in Fig. 1.

At the core of the SfmCAD is a two-stage learning strategy that operates in a coarse-to-fine manner to tackle the inherent complexity and time-intensiveness of simultaneous sketch profile and sweeping path learning. Initially, the network is trained to learn a coarse Box+Path representation of the shape, enabling a quick grasp of the shape’s path and broad structure using box-like geometric proxies. Then, an implicit network delves deeper into the geometric details of the shape, enhancing the reconstruction accuracy. We demonstrate the capability and versatility of our method by applying SfmCAD to many different types of objects, including CAD parts [16], ShapeNet objects [1], tree branches [20], and by comparing to previous works. We claim the following contributions:

1. We propose SfmCAD, a novel neural approach that parses 3D shapes as a set of industry-standard sketch-based CAD modeling operations. To the best of our knowledge, SfmCAD is the first unsupervised and universal neural network for learning the common CAD commands, including extrude, sweep, loft, and revolve.
2. We introduce novel decoupling of 3D structures and local geometric details, which are represented by 3D Bézier curves and implicit 2D sketches. This representation ensures accurate reconstruction and ease of editing.
3. We propose a two-stage learning strategy, namely 3D Box+Path learning and 2D implicit sketch learning, which significantly accelerates the training speed in shape parser learning.

2. Related work

**Geometric primitive extraction.** Geometric primitive is a commonly used representation for approximating and abstracting 3D shapes [13]. Traditional methods based on RANSAC [32], Hough transform [10], or variational optimization [4, 46] have been used to detect and fit primitives on point clouds or polygonal meshes, but they often require careful parameter tuning for each shape.

To overcome the overwhelming complexity of traditional methods, deep neural networks addressed the primitive segmentation/detection from point clouds [19, 47], they fit parametric surfaces [22, 29, 37], or detect parametric curves and sharp corners [9, 24, 40] to achieve compact CAD reconstructions. Although achieving remarkable accuracy on reconstruction tasks, these methods only output individual primitives with limited types, which restricts their capability for reconstructing complex and more general 3D shapes.

**Learning CSG-based reconstruction.** CSG reconstruction embodies a 3D shape of high complexity and non-convexity with CSG boolean operations to represent the 3D shape construction process [5]. CSGNet [36] first develops a neural model that parses a shape into a sequence of CSG operations. More recent works follow the line of CSG parsing by advancing the inference without any supervision [14] or improving representation capability with a three-layer re-formulation of the classic CSG-tree [30], or handling richer geometric and topological variations by introducing quadric surface primitives [49, 50]. However, CSG reconstruction tends to combine a large number of shape primitives that limit the user editing capabilities. Moreover, using only basic primitives (e.g., boxes, spheres) is insufficient for approximating complicated shapes, thus usually limiting the reconstruction accuracy of small details [50].

**Learning feature-based reconstruction.** Modern CAD workflows use feature-based modeling that decomposes a target shape into a sequence of CAD commands [34], such as drawing a sketch followed by CAD operations such as extrusion, loft, etc. This paradigm makes modeling more efficient and in tune with how designers and engineers work. Recent work has explored the potential of deep learning to generate 2D engineering sketches [7, 27, 33], or directly learn B-Rep CAD models [6, 11, 12, 17]. Several approaches propose generative models for CAD design, predicting a sequence of CAD modeling operations to produce editable CAD models, such as DeepCAD [42]. Fu-
Extrude Loft Sweep Revolve

Figure 2. Sketch-based feature modeling operators supported in SfmCAD. The resulting 3D shape of each operation is called a feature primitive.

Extrude Loft Sweep Revolve

3. Problem Statement and Overview

Sketch-based feature modeling creates 2D engineering sketches and then lifts them to 3D along a path using CAD operations [8, 21]. Here, we first introduce this typed sketch+path representation for sketch-based modeling and then give an overview of its generation from a voxel model.

3.1. Typed Sketch+Path Modeling

We define a profile as a closed curve, and its enclosed region is composed of one or multiple inner/outer loops. We define a sketch as the collection of one profile and its loops in a 2D plane. Let Z denote the sketch and C be a 3D path to extend the sketches to create a 3D solid shape. We define four principal operations that are commonly used in modern CAD workflow (see Fig. 2): Extrude extends Z along a linear path, Revolve produces a model by rotating Z around an axis, Sweep moves Z along a specified path to generate generalized cylinder shapes, and Loft allows for variations in the sketch, interpolating between multiple distinct sketches \( \{Z_1, Z_2, \ldots, Z_n\} \) along the shortest distances between them to create complex shapes.

Considering the similarity between the above operations, i.e., lifting 2D sketches to a 3D shape along a 3D path, we use a general typed sketch+path to represent all of these operations, where the term ‘typed’ indicates the specific type of operation. Without loss of generality, we call the 3D path a sweeping path and call the lofted or swept 3D shape a feature primitive. We further use a Bézier curve \( \Upsilon(t) \) to represent the sweeping path:

\[
C = \Upsilon(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i, \quad t \in [0, 1]
\]

where \( \mathbf{P}_i \) is a control point. The Bézier cubic is expressive enough to approximate the shapes shown in the paper, but higher-order curves can be obtained by simply changing the number of control points output by the network.

Moreover, it is noteworthy that the Extrude and Revolve operations can be regarded as general variants of the sweep operation because a Bézier curve becomes a linear path (for Extrude) if all the control points are collinear and an \( n \)-piece (usually \( n = 4 \)) cubic Bézier curve can approximate a circular path (for Revolve) [45]. In the following, we focus on reconstructing Sweep and Loft operations, where Sweep contains Extrude and Revolve, and the Loft is restricted to contain only two sketches with a linear path between them.

3.2. Overview

Given a 3D voxel shape, we aim to reconstruct the CAD model by neurally decomposing the target shape into a sequence of typed sketch+path representation, which can be executed by the corresponding CAD commands to assemble a compact CAD model. Learning the path and sketch of a shape simultaneously is challenging and time-consuming. Thus, we propose a two-stage learning approach to improve the efficiency of network training. First, the network learns a Box+Path representation to enclose the input shape, capturing the sweeping path, basic geometry, and rotation angles. Then, we learn implicit sketches that refine the coarse box+path into a detailed shape representation.

Fig. 3 details the two-stage learning network. We begin with a 3D convolutional encoder that transforms the voxel input into a feature vector \( \mathbf{z} \), which is then processed through a multilayer perceptron (MLP), yielding matrices \( \mathbf{P} \) and \( \mathbf{B} \). Here, \( \mathbf{P} \) are the control points of the Bézier curves representing the sweeping paths (Fig. 3 (b)), and \( \mathbf{B} \) encapsulates the box+path parameters (Fig. 3 (c)), including length, width, and sweep’s twist angles. Once the sweeping paths are derived from Stage 1, we focus on learning the geometric details surrounding each path. The feature \( \mathbf{z} \) is then mapped to \( N_p \) local features via another MLP. These vectors are subsequently concatenated with test points and fed into an implicit sketch network. The output from this network includes two occupancy values that represent the top and bottom sketches (Fig. 3 (d)), respectively. We then obtain the final CAD reconstruction by sweeping or lofting the detected sketches along the paths (Fig. 3 (e)).
4. Methodology

Here, we first present the box-path representation, which can be efficiently learned to approximate the input shape by stacking a set of box-like proxies along the sweeping path. Then, we define the neural sketch+path to refine the coarse reconstruction. Finally, we introduce the two-stage learning network for inferring this coarse-to-fine reconstruction.

4.1. Box+Path Representation

Previous unsupervised methods [23, 30, 31] for computing the Signed Distance Function (SDF) for geometric primitives (e.g., cubes, spheres, cylinders) rely on linear transformations that convert global testing point coordinates into local ones. However, the inherent non-linearity of curved paths limits the direct application of linear methods. We introduce a linear approximation method for the curved SDF computation, which we term Box+Path representation. We will first discuss the Box-Path in a 2D context to provide a more intuitive understanding before expanding it to 3D.  

**Box+Path in 2D.** Specifically, we approximate one feature primitive \( \mathcal{P} \) using a discrete set of \( N_s \) rigid box proxies, \( \{B_j\}_{j=1}^{N_s} \). The center position \( \mathbf{c}_i \) of each box \( B_i \) is obtained by uniformly sampling the Bézier curve \( \mathbf{Y}(i) \):

\[
\mathbf{c}_i = \mathbf{Y}(i/N_s), \quad i = 1, \ldots, N_s
\]

The height \( h_i \) and the normal vector (or heading direction) \( \mathbf{n}_i \) of each box are calculated as

\[
h_i = |\mathbf{c}_{i+1} - \mathbf{c}_i|, \quad \mathbf{n}_i = \frac{\mathbf{c}_{i+1} - \mathbf{c}_i}{|\mathbf{c}_{i+1} - \mathbf{c}_i|}
\]

By incorporating a learnable width parameter \( w \), the 2D Box+Path can be fully characterized.

**Box+Path in 3D.** When extended to 3D space, we introduce the tangent vector \( \mathbf{t}_i \) and the binormal vector \( \mathbf{b}_i \) to parameterize the orientation of the \( i \)-th box proxy. To allow the sketch to rotate by a prescribed angle along the sweeping path, we should learn rotation angles \( \delta^v \) and \( \delta^b \) at the start and end points of the sweeping path. Then, the intermediate rotation angles \( \delta_i \) are obtained by linear interpolation along the sweeping path. Next, we compute the tangent vector \( \mathbf{t}_i \) as the cross product of the normal \( \mathbf{n}_i \) and the positive z-axis unit vector \( \mathbf{e}_z \). We then rotate \( \mathbf{t}_i \) around \( \mathbf{n}_i \):

\[
\mathbf{t}_i = R_{\mathbf{n}_i}(\mathbf{t}_i \times \mathbf{e}_z, \delta_i),
\]

where \( R_{\mathbf{n}_i}(\mathbf{v}, \delta) \) represents a counter-clockwise rotation of vector \( \mathbf{v} \) with angle \( \delta \) around \( \mathbf{n}_i \). The binormal vector \( \mathbf{b}_i \) is obtained as a cross product of \( \mathbf{n}_i \times \mathbf{t}_i \). Altogether, the triplet \( \{\mathbf{n}_i, \mathbf{t}_i, \mathbf{b}_i\} \) defines the orientation for each box along the curve path. The 3D Box+Path is fully characterized by incorporating learnable dimension parameters \( l, w \).

4.2. Neural Sketch+Path

We present differentiable methods for lofting or sweeping 2D sketches into 3D solid shapes to accurately reconstruct.
the feature primitives. The shape of each 2D sketch is defined by an implicit representation, where we use \( \hat{S}_{sk} \) to represent the SDF of sketches. We now describe the differentiable operators that calculate the SDF of the 3D feature primitives corresponding to the sketches.

**Differentiable Sketch-Extrude.** In the context of generating the geometry of feature primitives from sketch and extrude operations, \( \hat{S}_{\text{extrude}} \) denotes the \( i \)-th sketch-extrude operation, \( x_{i} \) represents the \( z \)-component of the \( i \)-th position vector, and \( h_{i} \) is the height of the \( i \)-th feature primitive. We introduce a set of intermediate functions \( M_{1}, M_{2}, \) and \( M_{3} \) to facilitate a clearer formulation:

\[
M_{1}(\hat{S}_{sk}) = \max(C_{sk}, |x_{i} - h_{i}|),
M_{2}(\hat{S}_{sk}) = \max(\hat{S}_{sk}, 0),
M_{3} = \max(|x_{i} - h_{i}|, 0).
\]

The SDF of the \( i \)-th extrusion primitive is then given by:

\[
\hat{S}_{\text{extrude}}^{i} = \min(M_{1}(\hat{S}_{sk}), 0) + \sqrt{M_{2}(\hat{S}_{sk})^{2} + M_{3}^{2}}. \tag{5}
\]

**Differentiable Sketch-Loft.** For the loft operation, we interpolate between two distinct sketches, \( \hat{S}_{sk}^{u} \) and \( \hat{S}_{sk}^{l} \), at the upper and lower faces of the \( i \)-th feature primitive. Given the interpolation factor \( \alpha \) as \( |x_{i}/h_{i}| \), the linearly interpolated sketch, \( \hat{S}_{sk}^{\alpha} \), is computed as:

\[
\hat{S}_{sk}^{\alpha} = (1 - \alpha) \cdot \hat{S}_{sk}^{u} + \alpha \cdot \hat{S}_{sk}^{l}. \tag{6}
\]

We use the same intermediate variables as in the extrusion case, but with \( \hat{S}_{sk}^{u} \) replaced by \( \hat{S}_{sk}^{\alpha} \). The SDF of the \( i \)-th loft primitive can be expressed similarly to Eq. 5:

\[
\hat{S}_{\text{loft}}^{i} = \min(M_{1}(\hat{S}_{sk}^{\alpha}), 0) + \sqrt{M_{2}(\hat{S}_{sk}^{\alpha})^{2} + M_{3}^{2}}. \tag{7}
\]

**Differentiable Sketch-Sweep.** For sweep operation, we decompose it into a series of \( N_{s} \) extrusions. We denote the predicted SDF for the \( j \)-th extrude segment as \( S_{sk}^{j} \). The SDF of the \( i \)-th sweep primitive is then obtained by taking the minimum of the SDFs of all extrusion segments:

\[
\hat{S}_{\text{sweep}}^{i} = \min_{j=1,...,N_{s}} \hat{S}_{\text{extrude}}^{j}. \tag{8}
\]

**Transformation to Parametric CAD.** Following [23], we transform implicit sketches into parametric ones. With the control points of Bézier curves outputted by the network, we can achieve the sketch-path parametric reconstruction.

### 4.3. Two-stage Network Training

**Stage 1: Box-Path learning.** In this stage, the network predicts the control point coordinates \( \{P_{1},...,P_{4}\} \) of the cubic Bézier curve and the parameters \( \{l_{i}, w_{i}, \delta_{i}, \theta_{i}\} \) describing the \( i \)-th Box+Path \( B_{i} \). These parameters subsequently inform the computation of the remaining parameters \( \{h_{i}, n_{i}, t_{i}, b_{i}\} \), as detailed in Sect. 4.1. To align the path of \( B \) with the target shape path, we first compute the SDF, \( \hat{S}_{\text{box}}^{i,j} \), for each box at the \( i \)-th primitive and \( j \)-th segment, where \( i \in [1,N_{p}] \) and \( j \in [1,N_{s}] \), with \( N_{p} \) and \( N_{s} \) representing the total number of primitives and segments. Taking the union over the \( \hat{S}_{\text{box}}^{i,j} \) yields the SDF for the overall box+path \( \hat{S}_{B} \):

\[
\hat{S}_{B} = \bigcup_{i=1}^{N_{p}} \bigcup_{j=1}^{N_{s}} \hat{S}_{\text{box}}^{i,j}. \tag{9}
\]

where \( \bigcup \) is softmin(\( \mu \cdot x \)) and \( \mu \) is a hyperparameter. We then apply a differentiable converter \( \Psi \) similar to prior works [23, 31] to transform the \( \hat{S}_{B} \) into occupancy, yielding \( \hat{O}_{B} = \Psi(\hat{S}_{B}) \), where \( \Psi(x) = (\tanh(-\beta \cdot x) + 1)/2 \), and \( \beta \) is a hyperparameter. We then compute the Mean Squared Error (MSE) loss between \( \hat{O}_{B} \) and the ground truth occupancy \( \hat{O} \):

\[
L_{B} = \mathbb{E}_{x \in X} \left[ ||\hat{O}_{B} - \hat{O}||^{2} \right], \tag{10}
\]

where \( X \) is the set of all testing points. The exclusive use of \( L_{B} \) may lead to potential challenges: (1) the emergence of the curve with acute angles, and (2) the dimensions of \( B \) vastly outstrip the path length. Both increase the risk of shape self-intersections, which are considered invalid in CAD. To address these issues, we introduce two additional regularization terms,

\[
L_{\text{sm}} = \frac{1}{N_{p}} \sum_{i=1}^{N_{p}} (1 - \min_{j=1,...,N_{s}} (n_{i,j} \cdot n_{i,j+1})) \tag{11}
\]

\[
L_{lw} = \frac{1}{2N_{p}} \sum_{i=1}^{N_{p}} \text{ReLU}(h_{i} + w_{i} - 2 \cdot \Theta), \tag{12}
\]

where \( L_{\text{sm}} \) penalizes the sharpest curvature in the curve by maximizing the inner product between adjacent \( \mathbf{n} \), and \( L_{lw} \) penalizes any length or width greater than a threshold \( \Theta \). Combine with balance factors \( \lambda_{1}, \lambda_{2} \), the overall loss of Stage 1 is:

\[
L_{\text{box}} = L_{B} + \lambda_{1} L_{\text{sm}} + \lambda_{2} L_{lw}. \tag{13}
\]

**Stage 2: Implicit sketch learning.** Using the path learned in Stage 1, we progress towards fine reconstruction in Stage 2 by learning sketches. We initially utilize a Multi-Layer Perceptron (MLP) to convert the feature \( z \) into \( N_{p} \) local features \( z' \). These local features are then concatenated with the local \( \{x, y\} \) coordinates of each point and fed into an implicit sketch network, which is essentially an MLP predicting the SDF value \( \hat{S}_{sk} \). For the sketch-loft operation, two implicit sketch networks are used to learn \( \hat{S}_{sk}^{u} \) and \( \hat{S}_{sk}^{l} \). Following the computation methods outlined in Sect. 4.2, we can derive the predicted overall occupancy value \( \hat{O}_{F} \) by:

\[
\hat{O}_{F} = \begin{cases} 
\Psi(\bigcup_{i=1}^{N_{p}} \hat{S}_{\text{sweep}}^{i}), & \text{for sweep operation} \\
\Psi(\bigcup_{i=1}^{N_{p}} \hat{S}_{\text{loft}}^{i}), & \text{for loft operation} 
\end{cases}
\]
Finally, an MSE loss $\mathcal{L}_{rec}$ is computed against the ground truth $\tilde{O}$:

$$\mathcal{L}_{rec} = \mathbb{E}_{x \in \mathcal{X}} \left[ ||\hat{O}_T - \tilde{O}||^2_2 \right].$$  

(14)

The direct computation of $\mathcal{L}_{rec}$ over the entire point set $\mathcal{X}$ requires the repetition of $\mathcal{X}$ corresponding to $\hat{S}_{extrude}$ for $N_p \times N_s$ times, which is time-consuming. To accelerate the training of the implicit sketch network, we propose a real-time sampling strategy, exploiting the sketch consistency trait inherent in the sweep operation. Specifically, we sample grid-like testing points $y_i$ in the base sketch plane of the $i$-th path $\Upsilon_i$ obtained in the first stage. The occupancy value of $y_i$ is determined by inputting it into the implicit sketch network and applying the converter $\Psi$. Subsequently, $y_i$ is moved along the curve for $N_s - 1$ times, with its occupancy value directly replicated, yielding $\hat{O}_F^*$. The ground truth occupancy value $\tilde{O}^*_i$ is provided by the input voxel. After performing this procedure for all paths and conducting a union operation, the loss function corresponding to real-time sampling is given by:

$$\mathcal{L}^*_\text{rec} = \mathbb{E}_{y \in \Upsilon} \left[ ||\hat{O}_F^* - \tilde{O}^*_i||^2_2 \right].$$  

(15)

where $\Upsilon$ represents the set of testing points obtained through real-time sampling, and $\ast$ denotes computation using real-time sampling data.

Note that the SfmCAD is designed to choose between learning either loft or sweep operations, which is determined by its hyperparameters. However, enabling the network to concurrently learn both operations is also viable, with details provided in the supplementary materials.

5. Results and Evaluation

We evaluate the performance of SfmCAD on two reconstruction tasks: (1) holistic shapes of different categories and (2) semantically segmented shapes. We show the output editability of SfmCAD in the supplemental materials.

5.1. Setup

Implementation details. We implemented SfmCAD in PyTorch by using an NVIDIA® TITAN RTX GPU. The Adam optimizer [15] is utilized with a learning rate 1e-4 and beta values of (0.5, 0.99). The hyperparameters in Sect. 4.3 are set to: $\mu = 20$, $\beta = 50$, $\Theta = 0.1$, $\lambda_1 = 0.05$, $\lambda_2 = 0.05$. In our evaluations, we employ the best-performing operation (Loft/Sweep) for each category of holistic shapes and every part of segmented shapes. SfmCAD trains for 500 epochs per stage with a batch size of 24, followed by fine-tuning each test shape for 200 iterations per stage. All baseline methods are trained for 1,000 epochs and fine-tuned for 400 iterations for a fair comparison.

Evaluation metrics. We utilize three common metrics for 3D shape reconstruction: Chamfer Distance (CD), Normal Consistency (NC), and Edge Chamfer Distance (ECD). All experiments generate the reconstructed shapes via Marching Cubes at a resolution of $256^3$. To calculate CD, we sample 8,192 points from the ground truth and predicted shape surfaces. For ECD, we employ a threshold of 0.1 for normal cross-products to identify points close to the edges on the ground truth surface and 0.5 for the predicted surface, considering the lack of sharpness in shapes generated by Marching Cubes. For better visual clarity, we magnify the CD and ECD by factors of $10^3$ and $10^2$.

5.2. Reconstruction from Holistic Shape

We compare SfmCAD with UCSG-Net [14], CSG-Stump [30], ExtrudeNet [31] and SECAD-Net [23]. We test these methods on holistic shapes from different sources, including CAD parts in ABC [16], ShapeNet objects [1], and a synthetic tree dataset. The comparison against CAPRICNet [49] is provided in the supplementary materials.

Evaluation on ABC dataset. We employ the voxel grids and sampling point data provided by [49]. The dataset included 5,000 groups of data for training and 1,000 for testing. We select 100 shape subsets from the test set for eval-
Figure 6. Visual comparison on ShapeNet dataset.

Table 2. Quantitative comparison on ShapeNet dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CD↓</th>
<th>ECD↓</th>
<th>NC↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCSG-Net [14]</td>
<td>2.111</td>
<td>23.616</td>
<td>0.815</td>
</tr>
<tr>
<td>CSG-Stump [30]</td>
<td>2.168</td>
<td>17.592</td>
<td>0.818</td>
</tr>
<tr>
<td>ExtrudeNet [31]</td>
<td>0.910</td>
<td>14.110</td>
<td>0.849</td>
</tr>
<tr>
<td>SECAD-Net [23]</td>
<td>0.836</td>
<td>14.818</td>
<td>0.837</td>
</tr>
<tr>
<td>Ours</td>
<td>0.626</td>
<td>14.096</td>
<td>0.867</td>
</tr>
</tbody>
</table>

Table 3. Quantitative comparison on Tree dataset.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CD↓</td>
<td>7.242</td>
<td>2.197</td>
<td>0.686</td>
</tr>
<tr>
<td>NC↑</td>
<td>0.608</td>
<td>0.622</td>
<td>0.722</td>
</tr>
</tbody>
</table>

Evaluation on ShapeNet dataset. We utilize three categories from ShapeNet: chair, table, and display. The voxel grids and sampling points are provided by [3]. We selected 120 shapes for evaluation, with 40 from each category. Visual and quantitative results are in Fig. 6 and Tab. 2.

Evaluation on tree branches. We use a procedural modeling approach of Palubicki et al. [26] to generate tree meshes voxelized into a resolution of 64³. We adopt the method from [2] to obtain sampling points and occupancy values. This process results in 5,000 data groups, of which 4,500 are used for training and 500 for testing. We train all methods using increasingly denser voxel resolutions (16³, 32³, 64³) and select 50 trees for numerical evaluation. The visual results are shown in Fig. 7 and Tab. 3 for visual and quantitative results.

Summary of comparative analysis. As demonstrated in Tables 1, 2, and 3, SfmCAD consistently outperforms other methods in terms of CD, ECD, and NC, suggesting that the sketch+path representation of SfmCAD offers more flexible shape expression capabilities than the geometric primitives used by UCSG-Net and CSG-Stump, as well as the simple sketch+extrusion operations employed by ExtrudeNet and SECAD. Figures 5, 6, and 7 further show SfmCAD’s ability to generate detailed reconstructions (such as thin structures) across diverse shape categories.

5.3. Reconstruction from Segmented Shape

Semantic segmentation provides information for reconstructing complex shapes. Previous CAD reconstruction methods [23, 31] could not utilize this information because their single-type primitives lack the flexibility to represent each part independently. In contrast, SfmCAD successfully reconstructs multiple primitives with its sketch+path representation capability.

We show the task of unsupervised CAD reconstruction utilizing semantic segmentation. The evaluation is conducted on PartNet dataset [48], which provides instance-level 3D part information. We utilize three categories: chair, table, and trashcan. We voxelize the segmented parts and sampled occupancy values at resolutions 16³, 32³, 64³ following the scheme by [2]. This process yields a total of 12,000 chair parts (1,229 complete), 12,000 table parts (1,491 complete), and 3,000 trashcan parts (234 complete). The train-test split for all categories was set as 5:1. We selected 90 complete shapes, including 40 tables, 40 chairs, and 10 trashcans, for evaluation.

All methods were trained on part shapes using increas-
Figure 8. Visual comparison on PartNet dataset.

Table 4. Quantitative comparison on PartNet dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CD↓</th>
<th>ECD↓</th>
<th>NC↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtrudeNet [31]</td>
<td>3.251</td>
<td>0.549</td>
<td>0.783</td>
</tr>
<tr>
<td>SECAD-Net [23]</td>
<td>1.888</td>
<td>0.472</td>
<td>0.818</td>
</tr>
<tr>
<td>Ours</td>
<td>1.410</td>
<td>0.446</td>
<td>0.814</td>
</tr>
</tbody>
</table>

Table 5. Ablation study on sweep twist angles and loss terms.

<table>
<thead>
<tr>
<th>Settings</th>
<th>-δu, δl</th>
<th>-δl</th>
<th>-Llw</th>
<th>-Lsm</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD↓</td>
<td>4.092</td>
<td>3.567</td>
<td>4.804</td>
<td>3.268</td>
<td>3.142</td>
</tr>
<tr>
<td>ECD↓</td>
<td>0.173</td>
<td>0.139</td>
<td>0.164</td>
<td>0.154</td>
<td>0.133</td>
</tr>
<tr>
<td>NC↑</td>
<td>0.655</td>
<td>0.714</td>
<td>0.685</td>
<td>0.732</td>
<td>0.776</td>
</tr>
</tbody>
</table>

5.4. Ablation Study

We conducted an ablation study by using 40 box+path reconstructions on chair class from PartNet to evaluate the efficiency of our network design and loss terms. Initially, we disable the two sweep twist angles δu, δl or δl. Then we separately discarded Lsm and Llw. The visual results in Fig. 9 and the quantitative results in Tab. 5 suggest that removing any of these settings negatively affects the reconstruction performance of SfmCAD.

6. Conclusion and Future Work

We have presented an unsupervised neural network for CAD reconstruction that infers high-level CAD modeling operations. We have also introduced a unique sketch+path representation that encapsulates the local geometric details and overall structural information of the 3D shape. A novel two-stage learning strategy, operating in a coarse-to-fine manner, was proposed to address the complexity of simultaneous sketch and path learning, thus improving the efficiency of network training. We evaluated SfmCAD using several different categories of objects and demonstrated its advantages by comparing it to state-of-the-art methods.

In future work, we plan to explore using 2D sketches and 3D part templates to enhance the efficiency of unsupervised CAD reconstruction. Besides, given the manipulable property of SfmCAD’s outputs, extending this work for generative CAD design or modeling would be meaningful.

Acknowledgments. We thank the anonymous reviewers for their valuable suggestions. This work is partially funded by the Strategic Priority Research Program of the Chinese Academy of Sciences (XDB0640000), Guangdong Science and Technology Program (2023B1515120026), National Natural Science Foundation of China (62172416, 62172415, U22B2034, 62262043, 62365014), and Youth Innovation Promotion Association of the Chinese Academy of Sciences (2022131). This work was supported by NRCS grant #NR233A750004G044 to Benes. The findings and conclusions should not be construed to represent any agency determination or policy.
References


