ExtrudeNet: Unsupervised Inverse Sketch-and-Extrude for Shape Parsing

Daxuan Ren[1,2], Jianmin Zheng[✉,1], Jianfei Cai[1,3], Jiatong Li[1,2], and Junzhe Zhang[1,2]

[1] Nanyang Technological University, Singapore
[2] Sensetime Research
[3] Monash University
{daxuan001, asjmzheng, E180176, junzhe001}@ntu.edu.sg, jianfei.cai@monash.edu

Abstract. Sketch-and-extrude is a common and intuitive modeling process in computer aided design. This paper studies the problem of learning the shape given in the form of point clouds by "inverse" sketch-and-extrude. We present *ExtrudeNet*, an unsupervised end-to-end network for discovering sketch and extrude from point clouds. Behind ExtrudeNet are two new technical components: **1)** an effective representation for sketch and extrude, which can model extrusion with freeform sketches and conventional cylinder and box primitives as well; and **2)** a numerical method for computing the signed distance field which is used in the network learning. This is the first attempt that uses machine learning to reverse engineer the sketch-and-extrude modeling process of a shape in an unsupervised fashion. ExtrudeNet not only outputs a compact, editable and interpretable representation of the shape that can be seamlessly integrated into modern CAD software, but also aligns with the standard CAD modeling process facilitating various editing applications, which distinguishes our work from existing shape parsing research. Code is released at https://github.com/kimren227/ExtrudeNet.

1 Introduction

Pen draws a line, paint roller sweeps a surface, and pasta maker extrudes Fusilli from a stencil. From a point to a line, then to a surface and to a solid shape, the process of using lower dimensional shapes to construct a higher dimensional object seems to be a human instinct. In this paper, we explore the inverse of this process by training a neural network to infer 2D drawings of a point cloud and then extrude them into 3D to reconstruct the shape.

With recent development in 3D reconstruction technologies and cheaper sensors, point clouds can be easily obtained and become a widely adopted 3D data representation [23, 24, 34]. However, the unordered and unstructured nature of point clouds makes it difficult to perform high level manipulation and easy editing of their underlying geometries. Thus in recent years, the research of extracting shape features and generating high level shape representation from point clouds is very active, especially in computer vision and graphics.

We take inspiration from the process of sketch and extrude, a popular and intuitive approach widely used in the field of computer aided design (CAD) where engineers usually model shapes by first "sketching" a closed free form sketch (profile) in a 2D sketch plane and then "extruding" the sketch into 3D. We propose ***ExtrudeNet***, the first of its kind end-to-end unsupervised network for learning high level (editable and interpretable) shape representation through inverse sketch and extrude process from point clouds.

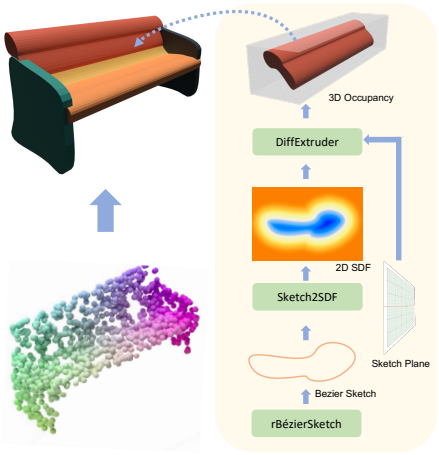

Fig. 1. ExtrudeNet studies the problem of learning the shape, given in the form of point clouds, by "inverse" sketch-and-extrude.

To realize ExtrudeNet, as shown in Fig. 1, we create three modular components: *1) rBézierSketch*, which generates a simple closed curve (i.e. no self-intersection); *2) Sketch2SDF*, a versatile numerical method for computing Signed-Distance-Field from parametric curves; and *3) DiffExtruder*, a differentiable method for extruding 2D Signed-Distance-Field (SDF) into a 3D solid shape. Built upon these components, ExtrudeNet takes a point cloud as input and outputs sketch and extrude parameters which form a compact, interpretable and editable shape representation. ExtrudeNet's outputs are highly compatible with modern CAD software [3], allowing control points based editing, which is much easier compared to directly editing triangle, polygonal meshes or even primitive based constructive solid geometry (CSG) models [26].

There are prior works on converting point clouds to high level shape representations. These representatives are discovering CSG in either a supervised or unsupervised manner. Supervised approaches [11, 19, 32, 35] suffer issues such as invalid syntax, infeasible models, and requiring large amount of expert annotated data. Unsupervised methods [8, 9, 21, 25, 33, 36] find the Boolean combinations of pre-defined geometric primitives such as box and cylinder. Our ExtrudeNet goes beyond these works. First, 2D sketch can be complex freeform curves, which allows us to model much more complex shapes using a single extrusion. Second, "Sketch-and-Extrude" is more user-friendly when it comes to editing and secondary-development, as editing a 2D sketch is more intuitive than editing 3D parameters. This Sketch-and-Extrude process happens to be a widely adopted method in CAD software for modeling 3D shapes [1–3, 15], making our method highly compatible with industry standards. Moreover, extensive experiments show that our ExtrudeNet can reconstruct highly interpretable and editable representations from point clouds. We also show through qualitative vi-

sualizations and quantitative evaluation metrics that ExtrudeNet outputs better overall results. The main contributions of the paper are:

- We present an end-to-end network, ExtrudeNet, for unsupervised inverse sketch and extrude for shape parsing. To the best of our knowledge, ExtrudeNet is the first unsupervised learning method for discovering sketch and extrude from point clouds.
- We design a special rational cubic Bézier curve based representation for sketch and extrusion learning, which can model freeform extrusion shapes, and the common cylinder and box primitives as well.
- We present a simple and general numerical method for computing the signed distance field of 2D parametric curves and their 3D extrusions which is proven to be suitable for gradient-based learning.

2 Related Work

Shape Representation. There have been different representations for 3D shapes. Recently implicit representation [12, 18, 20], usually in the form of Occupancy or Signed Distance Field, has drawn a lot of attention. It frees from intricate data representation and can be used directly via neural networks. To extract the underlying geometry, however, further processing is required [17], which is computationally intensive. Parametric representation describes shapes by parametric equations and is widely used in industry for modeling shapes thanks to its strong edibility and infinite resolution. However, generating parametric shapes from raw data like point clouds is a non-trivial task.

High Level Shape Learning. High level shape representations are often required, which benefit various practical applications. With the advance in machine learning, learning high level shape representations from raw data structure gains popularity. There have been many works for reconstructing CAD and especially CSG from point clouds. CSG is a tree-like structure representing shapes by starting from primitive objects and iteratively combining geometric shapes through Boolean operations [16]. It is well adapted in professional CAD software.

CSGNet [32] pioneers supervised CSG learning by modeling CSG as a sequence of tokens. It processes the sequence into a valid CSG-Tree using NLP techniques. With recent NLP technologies, DeepCAD [35] and CAD-As-Language [11] employ more powerful language models, e.g. Transformer, and add additional constraints to better predict CAD models. However, modeling CAD as language gives rise to addition problems (such as producing grammatically correct but invalid representations), which are not easy to solve.

VP [33] pioneers the unsupervised approaches by using the union of a set of boxes to approximate shapes. SQ [21] takes a step further by using super quadrics instead of boxes to better approximate complex shapes. BSPNet [8] and CVXNet [9] propose to use the union of a set of convexes to represent a complex shape, where the convexes are constructed by intersecting half-spaces. These methods extend the modeling capability of using standard primitives, but abundant planes are required to approximate freeform surfaces, which also limits

their edibility. UCSGNet [13] proposes CSG-Layers that iteratively selects primitives and Boolean operations for reconstruction. CSG-StumpNet [25] reformulates CSG-Tree with arbitrary depth into a fixed three layer structure similar to Disjunctive Normal Form, and uses a simple network to generate binary matrices for select fundamental products to union into the final reconstruction. However, these methods focuses on CSG operations and use only basic primitives (boxes, spheres, etc.), which is not efficient to approximate complicated shapes. CAPRI-Net [36] uses quadric implicit shapes to construct two intermediate shapes using an approach similar to BSP-Net. The two intermediate shapes are then subtracted to form the final shape. The construction process is interpretable, but the use of quadric implicit shapes reduces its edibility.

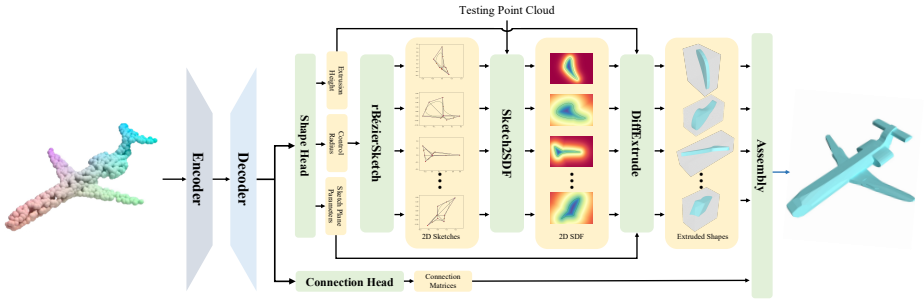

Fig. 2. Framework overview. The input point cloud first goes through the encoder-decoder phase to predict shape parameters and connection matrices via shape head and connection head. rBézierSketch is then used to generate the profile curve from the decoded sketch parameters. Sketch2SDF computes the SDF of the sketches. Sketch SDF, sketch plane parameters, and extrusion height are passed into DiffExtruder to generate the occupancy of the extrusion shapes, which together with the connection matrices are passed into CSG-Stump to generate the final shape occupancy.

3 ExtrudeNet

This section presents ExtrudeNet, an end-to-end network for unsupervised inverse sketch and extrude for compact and editable shape parsing. The input to the network is a point cloud representing a shape to be learned. ExtrudeNet outputs a set of 3D extrusions as the building blocks and their Boolean operations, which together create the shape. Each of the 3D extrusions is defined by a 2D sketch profile curve and a 3D extrusion process.

The pipeline of the entire network is illustrated in Fig. 2. The main components are briefly described below. Note that different encoder and assembly methods can be used to adapt for different use cases.

(1) **Encoder-Decoder:** ExtrudeNet first encodes the input point cloud into a latent feature using the off-the-shelf DGCNN as a backbone encoder [34]. The latent code is then enhanced by three fully connected layers with size 512, 1024, and 2048. After that, the latent feature is passed to the shape

head and the connection head to decode into extruded shape parameters and connection matrices. Extruded shape parameters consist of 2D sketch parameters, sketch plane parameters, and extrusion height. Connection matrices represent Boolean operations among the extruded shapes. Since binary value is not differentiable, we use the Sigmoid function to predict a soft connection weight in $[0, 1]$ for each matrix.

(2) **rBézierSketch:** rBézierSketch is used to convert sketch parameters into a closed profile curve defined by a set of rational cubic Bézier curves for extrusion. The generated 2D sketch curve is guaranteed to be free of self-intersection and can represent free-form curves, circular arc and even polygon in a single formulation.

(3) **SDF-Generation:** This consists of Sketch2SDF and DiffExtruder. Sketch2SDF is first used to compute the Signed-Distance-Field (SDF) of the generated 2D sketch on a plane. The computed 2D sketch SDF, the sketch plane parameters, and extrusion height are then passed into DiffExtruder to compute the extruded shapes' occupancy field in 3D space.

(4) **Assembly:** Given the predicted extrusion shapes' occupancy and connection matrices, we are in a position to assemble the extrusion shapes to complete the final reconstruction. For this purpose, we choose to directly use CSG-Stump from [25] for its simplicity and learning friendly nature. CSG-Stump reformulates CSG-Trees into a three layer structure similar to Disjunctive Normal Form and use three fixed size binary matrices to generate and select fundamental products. The first layer of CSG-Stump is a complement layer indicating whether the complement of the input occupancy should be used for the down steam operations. The second layer is an intersection layer which selects and intersects complement layer's outputs into intersected shapes. The last layer selects intersected shapes to union into the final shape.

Below we describe rBézierSketch and SDF-Generation in more detail.

3.1 rBézierSketch and Extrusion

To create an extrusion shape, a profile curve should be sketched and then extruded in 3D space. We use a network to predict the sketch parameters that define the profile curve in the $XY-$plane (serving as its local coordinate system) and then extrude it along the $Z-$direction to create the extrusion shape. The shape is then transformed to the required location and orientation predicted by the network which mimics the sketch plane transform in CAD software.

rBézierSketch. Considering the common modeling practice and shapes in CAD applications, there are a few assumptions for the profile curve: (i) it is closed and has no self-intersection, which has advantages in defining a valid solid shape; (ii) it is piecewise smooth for creating quality shapes; and (iii) it can model freeform curves, and circles or polygons as well. Meanwhile, we also have to balance the capability and complexity of the representation such that it can be easily

deployed into a learning pipeline. Based on these considerations, we propose the following model for our profile curve.

The basic mathematical model is a closed curve formed by N curve segments defined by special rational cubic Bézier curves $C_k(t), t \in [0,1], k = 0, 1, \cdots, N-1$, which may explain the name *rBézierSketch*. The equation of $C_k(t)$ is:

$$C_k(t) = \frac{P_0^k B_0^3(t) + w_1^k P_1^k B_1^3(t) + w_2^k P_2^k B_2^3(t) + P_3^k B_3^3(t)}{B_0^3(t) + w_1^k B_1^3(t) + w_2^k B_2^3(t) + B_3^3(t)} \tag{1}$$

where $P_i^k = (x_i^k, y_i^k)$ are the control points on the $XY-$plane, weights $w_1^k \geq 0, w_2^k \geq 0$ are for the two inner control points, and $B_i^3(t) = \binom{3}{i}(1-t)^{3-i}t^i$ are Bernstein polynomials. To make sure that consecutive segments are connected to form a closed curve, the constraints $P_3^k = P_0^{(k+1) \bmod N}$ are added.

The closed curve is defined around the origin. Thus it is convenient to express each control points $P_i^k = (x_i^k, y_i^k) = (\rho_i^k \cos(\alpha_i^k), \rho_i^k \sin(\alpha_i^k))$ by the radial coordinate ρ_i^k and the polar angle α_i^k. For simplicity, we further distribute the central angles of the segments evenly. That is, each Bézier curve has the central angle $\frac{2\pi}{N}$. Within each segment $C_k(t)$, the polar angles of control points are chosen to be:

$$\alpha_1^k = \alpha_0^k + \theta, \quad \alpha_3^k = \alpha_0^k + \frac{2\pi}{N}, \quad \alpha_2^k = \alpha_3^k - \theta \tag{2}$$

where $\theta = \dfrac{2\pi}{4N} + \tan^{-1}\left(\dfrac{1}{3}\tan\left(\dfrac{2\pi}{4N}\right)\right)$. It is worth pointing out that these polar angles are specially designed to achieve the capability of circle recovery (see Proposition 2 below). In this way, to specify the Bézier control points, we just need to provide the radial coordinates. Connecting all the control points in order forms a polygon that is homeomorphic to the origin-centered unit circle, which assures good behavior of the generated profile curve. In summary, the network only has to estimate the radial coordinates ρ_i^k and the weights w_1^k, w_2^k in order to sketch the profile curve.

Remark 1. The proposed curve model is specially designed to deliver a few nice properties, which are outlined in the following paragraphs.

Rational cubic Bézier representation in (1) is proposed because it is a simple form of NURBS that is the industry standard in CAD and meanwhile sufficient to model freeform curves [10]. Besides freeform smooth shapes, this representation is able to represent a straight line or a polygon, for example, as long as we let all the control points P_i^k lie on a line. The reason that the rational form is chosen is that it includes polynomial curves as a special case and has the capability of exactly representing a circle. These properties enable the extrusion shapes to include box and cylinder primitive shapes as special cases.

Due to the special angular set-up of rBézierSketch, the generated profile curve is a simple closed curve, i.e., it does not self-intersect. In fact, we have an even stronger result.

Proposition 1. *The area bounded by the curve generated by rBézierSketch is a star-shaped set.*

Moreover, the special choice of angle θ in (2) makes it possible to exactly represent a circular arc using Eq. 1.

Proposition 2. *Let the polar angles be given by Eq. 2. If* $\rho_0^k = \rho_3^k$, $\rho_1^k = \rho_2^k = \frac{\rho_0^k}{\cos(\theta)}$, *and* $w_1^k = w_2^k = \frac{1}{3}\left(1 + 2\cos\left(\frac{\pi}{N}\right)\right)$, *then the rational Bézier curve of (1) defines a circular arc, as shown in Fig. 3 (left).*

The proposed curve model assures C^0 continuity among the segments since the Bézier control polygons are connected in a loop by enforcing each curve's last control point to be the same as the next curve's first control point (see Fig. 3 (middle)). Thus totally $3N$ radial coordinates and $2N$ weights need to be predicted. In case the predicted control points and weights happen to satisfy certain condition given by Proposition 3, C^1 continuity can be achieved.

Proposition 3. *If the control points and weights satisfy*

$$P_0^{k+1} = P_3^k = \frac{\rho_2^k P_1^{k+1} + \rho_1^{k+1} P_2^k}{\rho_2^k + \rho_1^{k+1}}, \quad \frac{w_2^k}{w_1^{k+1}} = \frac{\rho_1^{k+1}}{\rho_2^k}, \tag{3}$$

curve segments $C_{k+1}(t)$ *and* $C_k(t)$ *meet at* P_0^{k+1} *with* C^1 *continuity.*

The proof of above propositions is provided in the supplementary material.

If for some CAD applications we already have prior knowledge that the models should be at least C^1 continuous, then we can enforce C^1 continuity by letting P_0^k and w_2^k be computed from Eq. 3. In this case we just predict $2N$ control points P_1^k, P_2^k and N weights w_2^k, which have fewer variables (see Fig. 3 (right)).

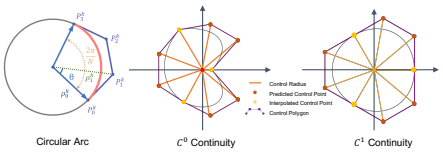

Fig. 3. Left: a circular arc. Middle: a C^0 profile curve. Right: a C^1 profile curve.

Remark 2. In CAD there have been some works to define single-valued curves in the polar coordinate system. For example, Sanchez-Reyes proposed a subset of rational Bézier curves that can be used to define single-valued curves [28] and later extended them to splines [29]. It should be pointed out that our proposed curves are different from those proposed by Sanchez-Reyes. Particularly, for a Sanchez-Reyes's curve, the control points are on radial directions regularly spaced by a constant angle and each weight must equal the inverse of the radial coordinate of the corresponding control point, which leaves very few degrees of freedom (DoF) for shape modeling. Our curves have more DoFs.

Extrusion. Once the profile curve $C(t) = (x(t), y(t))$ on the $XY-$plane is obtained, the extrusion shape is generated by directly extruding the curve along the $Z-$direction. The extrusion shape is bounded by the top and bottom planes and a side surface. The side surface has the parametric equations $(x, y, z) = ((x(t), y(t), s)$ where $s \in [0, h]$ is the second parameter and h is the extrusion height estimated by the network.

The network also estimates a quaternion that defines a rotation matrix R and a translation vector $\mathbf{t} = (t_x, t_y, t_z)$ that defines a translation matrix T. The matrix R makes the $XY-$plane be in the orientation of the target sketch plane and the matrix T moves the origin to the target sketch center on the sketch plane. Thus applying matrices R and T to the upright extrusion shape gives the target extrusion shape in 3D space, as shown in Fig. 4.

3.2 SDF-Generation

Sketch2SDF. Note that the generated sketches are in parametric forms. Unlike explicit or implicit functions, computing a signed-distance-field of parametric curves is not trivial. There were a few implicitization algorithms for converting a parametric curve into an implicit representation [30, 37]. However, they require exact arithmetic computation. Moreover, as observed in [31], the existence of singularity in parametric representation often makes the resulting implicit expression useless.

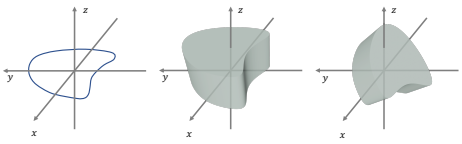

Fig. 4. Left: 2D sketch in the $XY-$plane; Middle: direct extrusion; Right: the target extrusion shape.

In this section, we present a numerical method for computing the SDF of parametric sketches. The method is general. While it applies to the rational cubic Bézier curves here, it also works for other parametric curves. This method also yields good gradients which is friendly for deep learning applications (see Sec. 4.2).

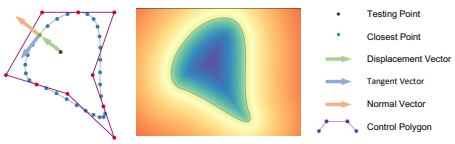

Fig. 5. The SDF of a curve is computed by a numerical method.

The SDF of a sketch is defined by the Distance-Field $DF(p)$, the smallest distance from a given testing point p to the curve, multiplied by a sign $SIGN(p)$ which indicates whether the testing point p is inside or outside of the sketch.

Distance-Field. To compute the Distance-Field of a sketch, we first sample a set of points S from the curve and take the smallest distance between the test point p and sample points S as the value of $DF(p)$. Specifically, given a parametric curve $C(t) = (x(t), y(t)), t \in [0, 1]$, the sampling points are obtained by evenly sampling the parameter values in the parameter domain:

$$S = \left\{ \left(x(\frac{i}{n}), y(\frac{i}{n}) \right) \mid i = 0, 1, \cdots, n \right\} \tag{4}$$

Then the distance between the testing point p and the set S is found by

$$DF(p) = \min_{s \in S} \|s - p\|_2 \tag{5}$$

and its corresponding closest point's parameter value is denoted by $CT(p)$.

Signed Distance-Field. To test if a point p is inside or outside of a sketch, we check whether if the displacement vector from the testing point towards the closest point in S has the same direction as the normal vector of the curve at the closest point. We assume that the curves are parameterized such that they are traced in the counter-clockwise direction. Otherwise, a simple reparameterization can correct it. Then the normal vector of the curve $C(t)$ can be computed by rotating its tangent vector by 90 degrees clockwise:

$$N(t) = \left(\frac{dy(t)}{dt}, -\frac{dx(t)}{dt}\right) \tag{6}$$

If two consecutive curve segments of the sketch meet with C^0 continuity, the normal vectors of the two segments at the joint point are different. To ensure a consistent normal at the junction of two curves, the average of the two is used. In this way, the sign function for a testing point p can be computed:

$$SIGN(p) = \frac{N(CT(p))) \cdot (C(CT(p)) - p)}{\|N(CT(p))) \cdot (C(CT(p)) - p)\|_2 + \epsilon} \tag{7}$$

where a small positive number ϵ is added to prevent from zero division. Finally, the Signed-Distance of p is:

$$SDF_s(p) = SIGN(p) \times DF(p). \tag{8}$$

Fig. 5 gives an example of a computed SDF.

DiffExtruder. Now we show how to compute the SDF of the extrusion shape. We first transform the testing point p back to a point p' by reversing the transformations that transform the $XY-$plane to the target sketch plane: $p' = R^{-1}(T^{-1}(p))$. Then we compute the signed distance of p' with respect to the upright extrusion shape whose base lies on the $XY-$plane. For this purpose, we project point p' onto the $XY-$plane by setting its z-value to 0 and denote the footprint by p'', and further find its nearest point $(c_x, c_y, 0)$ on the profile curve. Then the projected testing point p'', the nearest point $(c_x, c_y, 0)$ and its corresponding point (c_x, c_y, h) on the end plane of the extrusion define a vertical plane. Now, the problem of finding the SDF of the extrusion shape is reduced into computing the 2D SDF on the vertical plane.

If a point is inside of the extrusion shape, its signed distance to the shape simply becomes the minimum of the 2D sketch SDF and the distances from the point to the two planes that bound the extrusion.

$$SDF_i(p') = \max(\min(SDF_s((p'_x, p'_y)), h - p'_z, p'_z), 0) \tag{9}$$

If the point lies outside the extruded primitive, its signed distance becomes:

$$\begin{aligned} SDF_o(p') = -[&(\min(h - p'_z, 0))^2 + (\min(p'_z, 0))^2 \\ &+ (\min(SDF_s((p'_x, p'_y)), 0))^2]^{\frac{1}{2}}. \end{aligned} \tag{10}$$

It can be seen that for a testing point outside the extruded shape, SDF_i becomes zero, and for a testing point inside the extrusion shape, SDF_o becomes zero. Therefore we can simply get the overall SDF by adding the two terms:

$$SDF(p') = SDF_i(p') + SDF_o(p'). \tag{11}$$

Occupancy of Extrusion Shape. After getting the SDF of an extrusion shape, the occupancy can be computed by a sigmoid function Φ where η indicates how sharp the conversion is taken place:

$$O(p') = \Phi(-\eta \times SDF(p')). \tag{12}$$

3.3 Training and Inference

We train ExtrudeNet in an end-to-end and unsupervised fashion as no ground truth sketch and extrusion parameters are present. The supervision signals are mainly generated by the reconstruction loss L_{re} that computes the discrepancy between the reconstructed shape's occupancy $\hat{O}$ and the ground truth shape's occupancy O^* using a set of testing points $P \subset R^3$ sampled within the shape's bounding box:

$$L_{re} = \mathbb{E}_{p \sim P} ||\hat{O}_i - O_i^*||_2^2. \tag{13}$$

As suggested in [25], when a shape is too far from any testing points, the gradient becomes extremely small due to the use of a sigmoid function. Thus a primitive loss is introduced to "poll" the primitive shapes towards testing points:

$$L_{prim} = \frac{1}{K} \sum_{k}^{K} \min_{n} SDF_k^2(p_n), \tag{14}$$

We enforce the weights to be close to one, which encourage the rational Bézier curves to be close to Bézier curves.

Thus, the overall objective is then defined:

$$L_{total} = L_{re} + \lambda_p \cdot L_{prim} + \lambda_w \cdot \sum_{k=0}^{N-1} \sum_{i=1}^{2} (w_i^k - 1)^2 \tag{15}$$

where λ_p and λ_w are the trade-off factors.

During inference, we discretize the connection matrices into Boolean values for interpretable construction. In addition, we implement a CAD converter that directly takes binary connection matrices and the network generated parameters as input and converts the reconstructed shape into CAD compatible format. We use OpenSCAD [15] to render the final shape into STL [27] format.

4 Experiments

In this section, we first evaluate our complete pipeline ExtrudeNet, particularly its ability to extract editable and interpretable shape representations with ablation on different settings. Then, we evaluate the key component of rBézierSketch and Sketch2SDF on 2D toy examples.

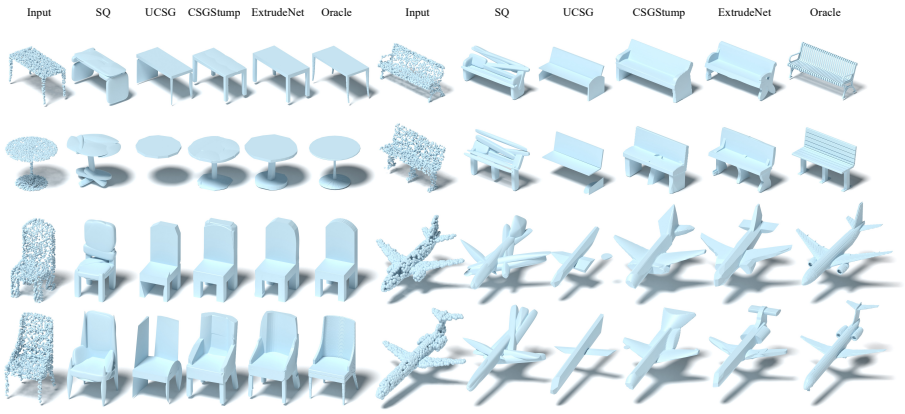

Fig. 6. Qualitative comparison between ExtrudeNet and other baselines. ExtrudeNet generates visually more pleasing outputs with curved surfaces compared to all the baselines.

4.1 Evaluations of ExtrudeNet

Dataset. We evaluate the ExtrudeNet on ShapeNet Dataset [7] with train, test, and val split aligned with prior methods. The input point clouds are sampled from the original mesh surface via Poisson Disk Sampling [6]. The testing points are sampled using [5] in a grid from the mesh bounding box with 15% of padding on each side. This padding is important to remove unwanted artifacts, see Supplementary Material.

Implementation Details. We implement ExtrudeNet using PyTorch [22], and train the network with Adam Optimizer [14] with a learning rate of 1e-4. We train each class on a single NVIDIA V100 GPU with a batch size of 16. It took about 5 days to converge.

In our experiments, each sketch is constructed with 4 Bézier curves with a sample rate of 100. We estimate 64 extruded shapes in total, and use 64 as the number of intersection nodes in Assembly (see Fig. 2) adopted from CSG-Stump [25].

Main Results - Comparisons with Baselines. We compare our method with related approaches focusing on editable shape abstraction, namely, VP [33], SQ [21], UCSG-Net [13], and CSG-Stump Net [25]. Note that BSPNet and CVXNet are excluded as they mainly focus on the reconstruction of thousands of planes that contain too many primitives and thus lack editability. For fair comparison, we align the number of available primitives to 64 for VP, SQ and UCSG. We show qualitative comparisons with the baseline methods in Fig. 6. It can be seen that our ExtrudeNet models curved surfaces more effectively compared to all the baseline methods, thus resulting in a more detailed reconstruction.

Moreover, we also conduct quantitative evaluations using symmetric L_2 Chamfer Distance (CD), volumetric IoU and surface F1 scores. The compar-

isons of the results are given in Table 1. We can see that ExtrudeNet achieves the best overall reconstruction quality.

Table 1. Different Metric computed between 3D reconstruction results and the ground truth shapes. ExtrudeNet outputs better overall results

	VP [33]			**SQ** [21]			**UCSG** [13]			**Stump** [25]			**ExtrudeNet**		
	CD	V-IoU	F1	CD	V-IoU	F1	CD	V-IoU	F1	CD	V-IoU	F1	CD	V-IoU	F1
Chair	**1.10**	0.28	**67.07**	2.85	0.21	41.49	3.54	0.25	61.01	1.34	**0.41**	63.33	1.47	**0.41**	60.17
Car	1.02	**0.67**	62.71	1.25	0.17	78.58	**0.64**	0.11	**82.54**	0.76	0.32	76.45	0.67	0.41	82.20
Sofa	2.18	0.29	48.75	1.27	0.37	69.14	1.30	0.29	79.59	0.85	0.61	**83.88**	**0.79**	**0.62**	83.42
Plane	5.11	0.29	37.55	**0.58**	0.23	82.94	0.71	0.10	**87.66**	0.70	**0.36**	74.57	0.66	0.33	77.95
Lamp	8.41	**0.24**	35.57	**1.79**	0.16	**63.00**	7.02	0.22	55.66	3.48	**0.24**	57.93	3.77	0.20	42.86
Telephone	2.75	0.55	48.22	0.46	0.38	88.29	**0.34**	**0.69**	**93.32**	1.60	0.58	91.24	0.56	0.62	89.00
Vessel	2.84	0.37	54.21	**0.76**	0.27	**78.52**	4.54	0.09	61.65	1.15	0.44	72.30	1.63	**0.48**	61.06
Loudspeaker	1.67	0.45	50.06	2.07	0.34	65.00	1.81	0.19	56.35	1.70	0.52	63.32	**1.21**	**0.60**	**75.04**
Cabinate	3.16	0.48	42.90	1.97	0.31	40.52	1.09	0.38	73.56	0.77	0.56	79.11	**0.73**	**0.68**	**84.94**
Table	1.62	0.26	60.05	2.89	0.17	60.01	3.64	0.30	65.09	1.35	0.35	**74.48**	**1.06**	**0.45**	73.49
Display	1.25	0.36	60.52	**0.72**	0.33	80.09	1.13	0.54	78.18	1.64	0.50	75.75	0.96	**0.55**	**83.66**
Bench	1.57	0.26	63.31	1.09	0.17	73.46	1.73	0.21	74.12	1.04	0.29	73.40	**0.78**	**0.39**	**81.17**
Rifle	1.35	0.35	66.06	**0.38**	0.26	**90.55**	1.05	0.29	84.38	0.78	0.37	85.61	0.76	**0.44**	84.18
Mean	2.61	0.37	53.61	1.39	0.25	70.12	2.19	0.28	73.31	1.32	0.42	74.64	**1.15**	**0.47**	**75.32**

Effect of Different Numbers of Primitives. To show that our proposed extrude shapes with freeform profile curves are highly adaptable, we train ExtrudeNet with limited numbers of extruded shapes. Table 2 reports the results on the three classes. It can be seen that even with only 8 available extruded shapes, our method can still achieve reasonable reconstruction results.

Table 2. CD results ($\times 10^{-3}$) with shapes extruded from different numbers of sketches.

#Primitives	8	16	32	64
Plane	0.89	0.96	0.81	**0.66**
Chair	2.01	1.81	1.47	**1.47**
Bench	1.41	1.18	1.30	**0.78**

Effect of Different Sketches. Our method can also use other profile curves such as polygonal and circular. Table 3 gives a comparison on the CD results under different types of profile curves. We can see that extruded shapes with the proposed freeform profile curves outperform the results using the other two common profile curves by a significant margin, which indicates the strong representability of the extruded shapes with freeform profile curves.

Table 3. CD results ($\times 10^{-3}$) with shapes extruded from different sketches on the Airplane class.

Sketch Type	Freeform	Circle	Poly
CD	**0.66**	0.877	0.838

Edibility of Sketch and Extrude. After getting the results from ExtrudeNet, a designer can do a secondary development with ease. Using the bench example in Fig. 7, we show that by editing the Bezier Control Points and reducing the extrusion height, we can easily generate an armchair from the reconstruction result.

More ablations such as sampling rate, number of curves, sketch plane placement etc., can be found in the supplementary material.

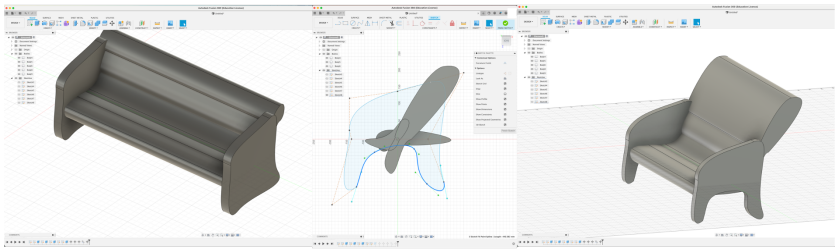

Fig. 7. As ExtrudeNet's output is directly compatible with Industrial CAD software, we can directly import the shape into Fusion360 and edit using its GUI interface. We show that a bench can be edited into an armchair by simply edit the 2D cross-section sketch by dragging the control points and adjust the extrusion height. **Left:** ExtrudeNet's output. **Middle:** Edit sketch using Fusion 360. **Right:** Result after editing.

4.2 Evaluation of rBézierSketch

To demonstrate the approximation ability and the ease of learning of the proposed rBézierSketch, we conduct a fitting experiment that directly optimize for the radial coordinate that best reconstructs a given raster emoji image from [4]. As the emojis contains different color blocks, which is not suitable to be represented using occupancy or SDFs, we choose to approximate the individual boundaries instead of solid colors blocks. Given a raster emoji image, we first compute its edges using Canny edge detection, and then we compute the Distance Field for the edge image. We fit the sketch by minimizing the MSE loss between the predicted and ground truth distance fields using standard gradient decent. As shown in Fig. 8, the emoji can be reconstructed by rBézierSketch with good quality. We also show the training plot, where a smooth decrease in the loss term indicating that rBézierSketch and Sketch2Field yields good gradients and it is friendly to used in a learning setup.

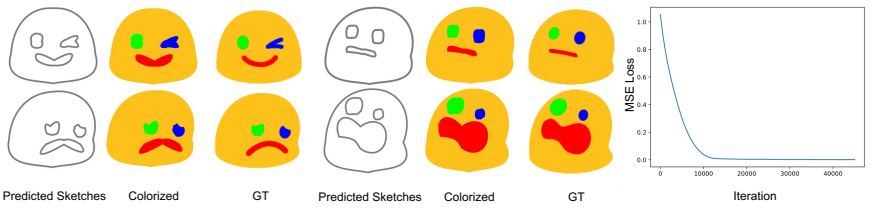

Fig. 8. Left: We show that using rBézierSketch alone we can reconstruct the emojis with good qualities. Right: Our rBézierSketch and Sketch2SDF provide low variance gradients which yields a smooth convergence.

4.3 Evaluation of Sketch2SDF

Here we demonstrate the versatility and accuracy of Sketch2SDF on computing parametric sketches' SDF by giving concrete examples.

Versatility. To show the versatility, we implemented four different kinds of parametric curves, namely polygon, ellipse, Cubic Bezier Sketchs with C^1 and C^0 continuity, see Fig. 9(Top). Note that our method is not limited to these curves, and can be easily adapted to new parametric curve types. During the implementation of these curves, only the curve sampling and normal computation need to be adapted accordingly while the rest of the code remains untouched. This indicates that our pipeline is highly reusable and can support new parametric curves with minimum effort.

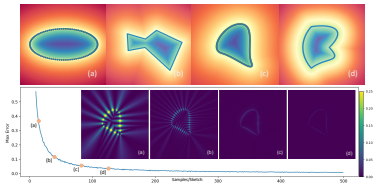

Fig. 9. Top: Sketch2SDF can be easily adapted to different parametric curve types. Bottom: Visualization of the approximation errors by comparing numerical and analytical SDFs of Bezier sketch. (a)∼(d) indicate the errors at sampling rate of 20, 40, 80 and 120. We can see that error is barely noticeable with a relatively small sampling rate.

Approximation Accuracy. As Sketch2SDF is a numerical method, approximation is introduced, where the sample rate is the most influential factor. Thus, we study the degree of approximation under different sampling rates by comparing our numerical results against the analytical results using Bézier Sketches. In Fig.9, we plot the maximum error of all testing points versus sample rate and visualize the error heat map. We can see that too few sample points will dramatically decrease the accuracy, and higher sampling rates yield more accurate approximation, which is however at the cost of higher computation and memory usage. Empirically, we find that around 400 sample points per curve is a reasonable trade-off.

5 Conclusion

In this paper, we have presented ExtrudeNet, which is an effective framework for unsupervised inverse sketch-and-extrude for shape parsing from point clouds. It not only output a compact, editable and interpretable shape representation but also a meaningful sketch-and-extrude process, which can benefit various applications. As demonstrated by extensive experiments, ExtudeNet can express complex shapes with high compactness and interpretability while achieving state-of-the-art reconstruction results both visually and quantitatively.

Discussion. Currently, ExtrudeNet requires a relatively long training time as sketches are more freeformed and have higher degrees of freedom. We also notice that some artifacts have been reconstructed due to high degrees of freedom. The current extrusion is a simple extrusion along the direction orthogonal to the sketch plane. Generalizing the extrusion to include more advanced processes such as sweeping and lofting warrants further investigation, which would have more impacts in the industry.

Acknowledgements The work is partially supported by a joint WASP/NTU project (04INS000440C130), Monash FIT Startup Grant, and SenseTime Gift Fund.

References

1. Shapr3d: The world's most intuitive 3d design app, https://www.shapr3d.com/ 2
2. Solidworks: 3d cad design software, https://www.solidworks.com/ 2
3. Fusion 360: Cloud powered 3d cad/cam software for product design (Jun 2021), https://www.autodesk.com.sg/products/fusion-360/overview 2
4. Color and black-and-white noto emoji fonts, and tools for working with them. (Mar 2022), https://github.com/googlefonts/noto-emoji 13
5. Batty, C.: Sdfgen. https://github.com/christopherbatty/SDFGen (2015) 11
6. Bridson, R.: Fast poisson disk sampling in arbitrary dimensions. SIGGRAPH sketches **10**, 1 (2007) 11
7. 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) 11
8. 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) 2, 3
9. 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) 2, 3
10. Farin, G.: Curves and Surfaces for CAGD: A Practical Guide. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edn. (2001) 6
11. Ganin, Y., Bartunov, S., Li, Y., Keller, E., Saliceti, S.: Computer-aided design as language. arXiv preprint arXiv:2105.02769 (2021) 2, 3
12. 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) 3
13. Kania, K., Zieba, M., Kajdanowicz, T.: Ucsg-net–unsupervised discovering of constructive solid geometry tree. arXiv preprint arXiv:2006.09102 (2020) 4, 11, 12
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 11
15. Kintel, M., Wolf, C.: Openscad. GNU General Public License, p GNU General Public License (2014) 2, 10
16. Laidlaw, D.H., Trumbore, W.B., Hughes, J.F.: Constructive solid geometry for polyhedral objects. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques. pp. 161–170 (1986) 3
17. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987) 3
18. 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) 3
19. Para, W.R., Bhat, S.F., Guerrero, P., Kelly, T., Mitra, N., Guibas, L., Wonka, P.: Sketchgen: Generating constrained cad sketches. arXiv preprint arXiv:2106.02711 (2021) 2
20. 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) 3

21. 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) 2, 3, 11, 12
22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf 11
23. 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) 1
24. 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) 1
25. 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) 2, 4, 5, 10, 11, 12
26. Requicha, A.A., Voelcker, H.B.: Constructive solid geometry (1977) 2
27. Roscoe, L., et al.: Stereolithography interface specification. America-3D Systems Inc **27**(2020), 10 (1988) 10
28. Sanchez-Reyes, J.: Single-valued curves in polar coordinates. Computer-Aided Design **22**(1), 19–26 (1990) 7
29. Sánchez-Reyes, J.: Single-valued spline curves in polar coordinates. Computer-aided design **24**(6), 307–315 (1992) 7
30. Sederberg, T.W., Anderson, D.C., Goldman, R.N.: Implicit representation of parametric curves and surfaces. Computer Vision, Graphics, and Image Processing **28**(1), 72–84 (1984) 8
31. Sederberg, T.W., Zheng, J., Klimaszewski, K., Dokken, T.: Approximate implicitization using monoid curves and surfaces. Graph. Model. Image Process. **61**(4), 177–198 (1999), https://doi.org/10.1006/gmip.1999.0497 8
32. 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) 2, 3
33. 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) 2, 3, 11, 12
34. 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) 1, 4
35. Wu, R., Xiao, C., Zheng, C.: Deepcad: A deep generative network for computer-aided design models. arXiv preprint arXiv:2105.09492 (2021) 2, 3
36. 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) 2, 4

37. Zheng, J., Sederberg, T.W.: A direct approach to computing the μ-basis of planar rational curves. J. Symb. Comput. **31**(5), 619–629 (2001). https://doi.org/10.1006/jsco.2001.0437, https://doi.org/10.1006/jsco.2001.0437 8