

POINT2PRIMITIVE: CAD RECONSTRUCTION FROM POINT CLOUD BY DIRECT PRIMITIVE PREDICTION

Anonymous authors

Paper under double-blind review

ABSTRACT

Recovering CAD models from point clouds requires reconstructing their topology and sketch-based extrusion primitives. A dominant paradigm for representing sketches involves implicit neural representations such as Signed Distance Fields (SDFs). However, this indirect approach inherently struggles with precision, leading to unintended curved edges and models that are difficult to edit. In this paper, we propose Point2Primitive, a framework that learns to directly predict the explicit, parametric primitives of CAD models. Our method treats sketch reconstruction as a set prediction problem, employing an improved transformer-based decoder with explicit position queries to directly detect and predict the fundamental sketch curves (i.e., type and parameter) from the point cloud. Instead of approximating a continuous field, we formulate curve parameters as explicit position queries, which are optimized autoregressively to achieve high accuracy. The overall topology is rebuilt via extrusion segmentation. Extensive experiments demonstrate that this direct prediction paradigm significantly outperforms implicit methods in both primitive accuracy and overall geometric fidelity.

1 INTRODUCTION

Computer-Aided Design (CAD) reconstruction, which aims to convert unstructured data like point clouds into structured, parametric models, is a long-standing goal in computer graphics and engineering (Kruth and Kerstens, 1998). Among various modeling techniques, the sketch-and-extrude process is one of the most fundamental operations, capable of generating a vast array of mechanical parts and industrial designs (Willis et al., 2021). Therefore, automatically reversing this process, transforming a raw 3D scan into a structured sketch-based CAD representation, holds immense value for digital manufacturing, legacy part recreation, and design modification.

Despite recent progress driven by large-scale datasets (Willis et al., 2021; Koch et al., 2019), a critical challenge lies in accurately capturing the model’s boundaries. The unstructured, noisy nature of point clouds contrasts sharply with the precise, topologically-defined structure of CAD models. A dominant paradigm in recent literature has been to represent 2D sketches using implicit neural representations such as Signed Distance Fields (SDFs) (Uy et al., 2022; Ren et al., 2022; Li et al., 2023a). While these methods are powerful for capturing complex, continuous surfaces, they remain limited in several critical aspects. First, SDFs are purely geometric fields that lack semantic structure, offering no direct correspondence to the primitive entities used in CAD. Second, as approximation-based representations, they inevitably introduce noise and blur fine geometric details, which makes it difficult to recover sharp corners or well-defined curves. Consequently, reconstructed sketches often exhibit **smoothed or rounded edges** rather than precise boundaries. Finally, because SDF-based sketches cannot be directly expressed in terms of editable primitives—**lines, circles, and arcs**, the lingua franca of CAD software—an additional conversion step is required. This transformation not only complicates the workflow but also accumulates parameter errors, further hindering faithful reconstruction and downstream editability.

The above limitations lead us to a fundamental question:

*Can we bypass the imprecise, intermediate representations and **directly** predict the explicit, parametric sketch primitives from the point cloud?*

In this work, we affirmatively answer this question by proposing *Point2Primitive*, a framework grounded in a direct prediction methodology for CAD reconstruction from point clouds. Unlike indirect approaches that rely on intermediate surface fitting or voxelization, *our method directly targets the underlying parametric sketch primitives*. Specifically, we observe that many CAD models can be systematically deconstructed into a sequence of extrusion operations, each defined by a two-dimensional sketch—a closed profile formed by geometric primitives such as lines, arcs, and circles—together with an associated three-dimensional operation that specifies the extrusion direction and length.

Building on this insight, we reformulate the complex reverse engineering problem into two interdependent sub-tasks. First, in the stage of *topology reconstruction*, the input point cloud is partitioned into clusters that correspond to individual extrusions, a process we term *extrusion segmentation*. This step disentangles the overall geometry into meaningful sub-structures, enabling subsequent sketch prediction to operate on localized and semantically coherent point sets. Second, in the stage of *primitive prediction*, we recover the generating two-dimensional sketch for each segmented extrusion body. To achieve this, we design an improved transformer-based decoder that refines point features and directly decodes them into parametric primitives. The decoder integrates explicit geometric priors by formulating curve parameters as positional queries, which guide the attention mechanism and progressively update predictions across layers. This design enhances the accuracy and stability of primitive recovery.

Point2Primitive could bridge topology recovery and parametric sketch inference within a unified direct-prediction framework. By directly predicting the language of CAD, our method produces CAD models that are not only geometrically accurate but also truly editable. In summary, our main contributions are summarized as follows:

- We introduce Point2Primitive, a new pipeline that *directly* predicts parametric sketch primitives (i.e., lines, arcs, and circles) from raw point clouds, bypassing implicit fields and preserving sharp, semantically meaningful boundaries for downstream editability.
- We decompose reconstruction into (i) *extrusion segmentation* to recover topology by clustering the point cloud into extrusion bodies, and (ii) *primitive prediction* with an improved transformer decoder that uses explicit position queries and autoregressive parameter refinement to yield high-precision primitive types and parameters.
- Extensive experiments and ablations show consistent gains in boundary fidelity, primitive recovery, and editability over SDF-based and recent learning baselines, demonstrating both accuracy and robustness across diverse CAD geometries.

2 RELATED WORKS

2.1 CAD RECONSTRUCTION VIA PRIMITIVE DETECTION

Several learning-based approaches have also been proposed to fit geometric primitives to point clouds. These methods first segment the input points into clusters corresponding to the same surface primitives and fit the clustered points to the parametric surface primitives (Li et al., 2019; Sharma et al., 2020; Le et al., 2021; Li et al., 2023b) or free surfaces (Liu et al., 2023). Except for surface primitives, UCSG-Net (Kania et al., 2020) and CSG-stump (Ren et al., 2021) learn the CSG parse tree that interrelates the predicted geometry primitives to reconstruct shapes. The reconstructed CAD models are in B-rep format, which is not as easy to edit as the sketch-and-extrude modeling procedure. In this paper, we explore the inverse of the sketch-and-extrude process and strive to recover the CAD modeling procedure from the point cloud. Therefore, we segment the point clouds into clusters belonging to the same extrusion primitive and explicitly reconstruct every element of the extrusion primitives. Instead of fitting points to the parametric or free surface, we employ an improved transformer to learn and predict the curves directly.

2.2 SKETCH-AND-EXTRUDE CAD RECONSTRUCTION

Some new solutions use SDF to represent sketches following the recent implicit 3D shape representations (Park et al., 2019; Wang et al., 2022). ExtrudeNet (Ren et al.) and SECAD-Net (Li et al.,

2023a) can learn implicit sketches and differentiable extrusions. Point2Cyl reconstructs 3D shapes through sketch regression supervised by the latent embeddings of its SDF. However, these methods utilize the implicit fields for sketch representation, leading to curved edges of the reconstructed shapes. Meanwhile, the reconstructed sketch is hard to be directly edited because it is represented by the SDF. Further transformation from SDF to sketch further brings in extra parameter errors.

2.3 CAD GENERATION

Early attempts explored CAD sequence generation with VAE-based models (Wu et al., 2021; Kingma and Welling, 2022) and their extensions such as vector-quantized VAE (Xu et al., 2023) or multimodal diffusion frameworks (Ma et al., 2024). However, tokenizing CAD sequences often disrupts the underlying CSG structure (Yu et al., 2021), making it difficult for language models to capture topology. While SkexGen (Xu et al., 2022) addressed this by separately encoding topology and geometry, existing VAE-LM hybrids still generalize poorly to reconstruction tasks. Moreover, LLMs struggle with numerical precision (Yang et al., 2024), leading to parameter inaccuracies and limited transferability beyond training data.

3 METHOD

3.1 PRELIMINARIES

A 2D CAD sketch can be modeled as a finite sequence of geometric primitives $S_i = (c_1^{(i)}, \dots, c_{N_i}^{(i)})$, where each primitive $c_j^{(i)} = (t_j^{(i)}, \rho_j^{(i)})$ comprises a type $t_j^{(i)} \in \mathcal{T}$ and a parameter vector $\rho_j^{(i)} \in \mathbb{R}^6$. We adopt three primitive types $\mathcal{T} = \{\text{L}, \text{A}, \text{C}\}$ for *line*, *arc*, and *circle*, respectively. Given the point set of the i -th extrusion, $\mathbf{P}_{\mathcal{E}_i} \in \mathbb{R}^{N_{\mathcal{E}_i} \times 3}$, the network f_θ is trained to predict the sketch as a set/sequence of primitives:

$$f_\theta(\{\mathbf{p}_k \mid \mathbf{p}_k \in \mathbf{P}_{\mathcal{E}_i}\}) \approx S_i = (c_1^{(i)}, \dots, c_{N_i}^{(i)}), \quad c_j^{(i)} = (t_j^{(i)}, \rho_j^{(i)}), \quad t_j^{(i)} \in \{\text{L}, \text{A}, \text{C}\}. \quad (1)$$

Center-prior parameterization. We use a center-prior parameterization with a fixed 6-D vector $\rho = (\rho_1, \dots, \rho_6)$ and pad unused entries by -1 . *Line* (L): midpoint $\mathbf{m} = (x_m, y_m)$ and one endpoint $\mathbf{d}_1 = (x_1, y_1)$, encoded as $\rho = (x_m, y_m, x_1, y_1, -1, -1)$. *Arc* (A): midpoint $\mathbf{m} = (x_m, y_m)$, start point $\mathbf{s} = (x_1, y_1)$, and end point $\mathbf{d}_2 = (x_2, y_2)$, encoded as $\rho = (x_m, y_m, x_1, y_1, x_2, y_2)$. *Circle* (C): center $\mathbf{c} = (x_c, y_c)$ and radius r , encoded as $\rho = (x_c, y_c, r, -1, -1, -1)$. Unless otherwise stated, we refer to a primitive by $c = (t, \rho)$. The first two entries of ρ (i.e., the center or midpoint) are further used as positional embeddings to guide attention in the improved transformer decoder (see Sec. 3.4).

Sketch-extrude-shape relation. An extrusion is denoted by $\mathcal{E}_i = [S_i, E_i]$, where E_i is the associated operation. A solid shape is a finite collection of extrusions, $\mathbb{S} = \{\mathcal{E}_1, \dots, \mathcal{E}_K\}$, and is represented by the signed distance field (SDF) induced by these extrusions. In our pipeline, $\{\mathcal{E}_1, \dots, \mathcal{E}_i\}$ is recovered via point segmentation, while the current sketch S_i is predicted with the improved transformer decoder.

Based on the above preliminaries, we present Point2Primitive, a direct sketch primitives prediction method for CAD reconstruction from point clouds. As shown in Figure 1, our method first segments points into extrusions, classifies base/barrel points, and then applies an improved transformer to predict curve types and parameters as a set-to-set problem. The segmentation head builds on PointCNN with fixed grid points, while the transformer refines features and enhances prediction accuracy.

3.2 CAD DATASET GENERATION

In this paper, we propose to generate the training dataset based on the Signed Distance Field (SDF). An SDF $\text{SDF}(p_{\text{eval}}, b)$ is a continuous function that, for a given spatial eval point p_{eval} , outputs the eval point’s distance to the closest boundary defined by b , whose sign denotes whether the point is inside (negative) or outside (positive) of the watertight surface. The data generation pipeline can convert any CAD sequence saved in JSON files following the style of the Fusion360 Gallery dataset (Willis et al., 2021) to the training data pairs.

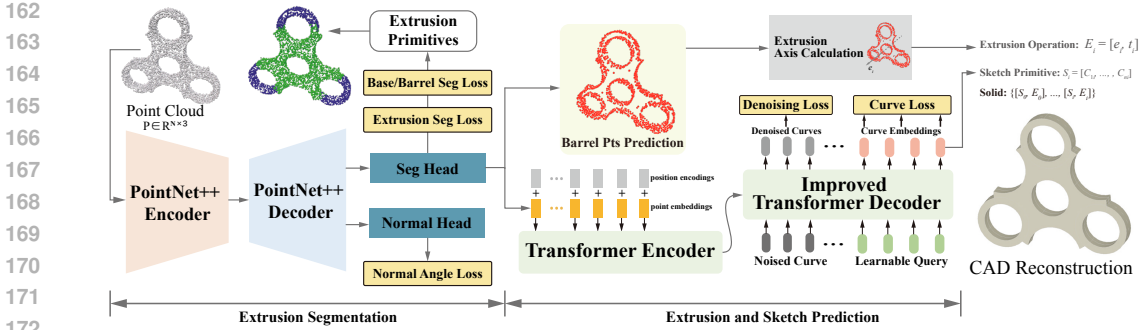


Figure 1: **Overview of Point2Primitive.** Our model consists of two main parts: extrusion segmentation and sketch prediction. In extrusion segmentation, a PointNet++ model is used to segment the barrel points and extrusions from the given point cloud to rebuild the topology. In sketch prediction, the point features from the Seg head are first refined using a vanilla transformer encoder and decoded into curves through the improved transformer decoder.

Given an extrusion point set $\mathbf{P}_{\mathcal{E}_i} \in \mathbb{R}^{N_{\mathcal{E}_i} \times 3}$ and its sketch S_i . We first project the points $\mathbf{P}_{\mathcal{E}_i}$ to the sketch plane as eval points according to the transform matrix W_i defined by its extrusion axis e_i . Then, its SDF is calculated to the boundary defined by the sketch S_i . The batch SDF calculation will be detailed in the Appendix F.1. The barrel label b_i of each point can be defined as follows:

$$b_i = \text{Boo1}(\text{SDF}(W_i \cdot \mathbf{P}_{\mathcal{E}_i}, S_i) \leq \text{thresh}) \quad (2)$$

where thresh controls the label precision and thresh = 0.01 in this paper.

As for the Extrusion Label, each extrusion mesh is first built using pythonOCC (Paviot, 2022) by the CAD sequence. Then, the SDF of the input points to each extrusion boundary defined by its mesh is calculated. The extrusion label of each point is set as its closest extrusion mesh.

3.3 EXTRUSION SEGMENTATION

To decompose the raw point cloud into semantically meaningful extrusion bodies, we design an extrusion segmentation network that clusters points belonging to the same extrusion. As illustrated in Figure 1, the segmentation module is implemented using a PointNet++ encoder–decoder backbone (Qi et al., 2017), followed by two prediction heads.

Specifically, the encoder extracts hierarchical features from the input point cloud, while the decoder progressively recovers point-wise embeddings. On top of these embeddings, two parallel branches are employed: the *Normal Head* predicts point-wise surface normals, and the *Extrusion Segmentation Head* predicts extrusion-related classes. The Ext Head outputs logits $\hat{\mathbf{M}}$ corresponding to $2K$ categories, where each extrusion is represented by two complementary channels. For the j -th extrusion, the $(2j)$ -th channel encodes the extrusion body assignment, whereas the $(2j + 1)$ -th channel encodes its associated barrel points. In this way, extrusion logits $\hat{\mathbf{W}}$ (for generating the extrusion \mathbf{p}_k) and barrel logits $\hat{\mathbf{B}}$ (for defining the the boundary b) can be computed as:

$$\hat{\mathbf{W}}_{:,j} = \hat{\mathbf{M}}_{:,2j} + \hat{\mathbf{M}}_{:,2j+1}, \quad (3)$$

$$\hat{\mathbf{B}}_{:,0} = \sum_j \hat{\mathbf{M}}_{:,2j}, \quad \text{and} \quad \hat{\mathbf{B}}_{:,1} = \sum_j \hat{\mathbf{M}}_{:,2j+1}. \quad (4)$$

Through this design, the network jointly captures extrusion topology and barrel boundary information, which facilitates accurate decomposition of the complex shape into extrusion units. Please refer to Appendix E for more implementation detail.

3.4 IMPROVED TRANSFORMER DECODER

To find the implicit design features encoded in the point features, we propose an improved transformer decoder to convert the refined point features into curves and directly predict the parameters. As shown in Figure 2, the curve embeddings encode the type information, and the curve parameters

are formulated as positional queries. By decomposing the learnable queries into type and parameter embeddings, explicit geometric priors are introduced, avoiding the abstract learning process and leading to high accuracy.

Parameter as Position Encoding. The curve parameters are formulated in a center-prior format detailed in Section 3.1, so the curve parameters can be formulated as position encoding \mathbf{PE}_i to guide the attention. Given the curve parameters ρ_j^i as the parameters of the j -th curve in the i -th sketch S_i . The corresponding positional queries \mathbf{PE}_j^i are generated by:

$$\mathbf{PE}_j^i = \text{MLP}(f_{pe}(\rho_j^i)) = \text{MLP}(f_{pe}(x_{m_j}^i, y_{m_j}^i)), \quad (5)$$

where f_{pe} is the function that generates the sinusoidal embeddings from the curve center. Multi-layer perceptron (MLP) and ReLU activations are used to produce the position encodings \mathbf{PE}_j^i .

Direct Curve Parameter Prediction. The curve prediction procedure can be seen as feeding the positional queries (curve parameters) and curve embeddings (type queries) into the decoder to probe the curves c_j^i that correspond to the position encodings while having similar patterns with the content in the point features. Furthermore, we propose to predict the curve parameters directly. The curve parameters are updated layer-by-layer in an autoregressive way to fit the target, as shown in Figure 2. The layer-by-layer update is based on the decoder output as follows:

$$\delta x_{m_j}^i, \delta y_{m_j}^i, \delta p_{2_j}^i, \delta p_{3_j}^i, \delta p_{4_j}^i, \delta p_{5_j}^i = \text{MLP}(O_n), \quad (6)$$

where O_n is the output of the n -th decoder layer. The curve parameters are updated to $[x_{m_j}^{i'}, y_{m_j}^{i'}, p_{2_j}^{i'}, p_{3_j}^{i'}, p_{4_j}^{i'}, p_{5_j}^{i'}]$ in each layer, as shown in Figure 2. All intermediate parameter outputs are supervised by ground truth. We further utilize query denoising (Li et al.) to accelerate the convergence.

3.5 TRAINING OBJECTIVE

The Point2Primitive is trained using a multi-task objective composed of extrusion segmentation loss \mathcal{L}_{ext} , base-barrel classification loss \mathcal{L}_{bb} , normal $\mathcal{L}_{\text{norm}}$ and sketch \mathcal{L}_{skh} prediction losses:

$$\mathcal{L} = \mathcal{L}_{\text{ext}} + \mathcal{L}_{\text{bb}} + \mathcal{L}_{\text{norm}} + \mathcal{L}_{\text{skh}}, \quad (7)$$

where the cross entropy loss is used in \mathcal{L}_{ext} and \mathcal{L}_{bb} while L-1 loss is used in $\mathcal{L}_{\text{norm}}$. The sketch prediction loss \mathcal{L}_{skh} can be formulated as:

$$\begin{aligned} \mathcal{L}_{\text{skh}} &= \sum_{i=1}^{N_C} \mathcal{L}_{\text{focal}}(\hat{t}_i, t_i) + \beta \sum_{i=1}^{N_C} \sum_{j=1}^{N_p} \mathcal{L}_p(\hat{\rho}_{i,j}, \rho_{i,j}), \\ \mathcal{L}_p &= \mathcal{L}_1(\hat{\rho}_{i,j}, \rho_{i,j}) + \mathcal{L}_2(\hat{\rho}_{i,j}, \rho_{i,j}), \end{aligned} \quad (8)$$

where $\mathcal{L}_{\text{focal}}(\cdot, \cdot)$ represents the focal loss (Lin et al., 2017) and $\mathcal{L}_1(\cdot, \cdot)$ and $\mathcal{L}_2(\cdot, \cdot)$ represents the L1 loss and L2 loss, respectively. Both denoising loss and curve Loss use the sketch prediction loss \mathcal{L}_{skh} , as shown in Figure 1. More specifically, in the L2 loss, we randomly select n points on each curve and compute the chamfer distance as the L2 loss. N_p is the number of parameters ($N_p = 6$ is this paper) while $N_C = 30$ is the number of curves. β is the weight to balance both terms ($\beta = 2$ in this paper) and $\hat{\cdot}$ denotes the ground truth value.

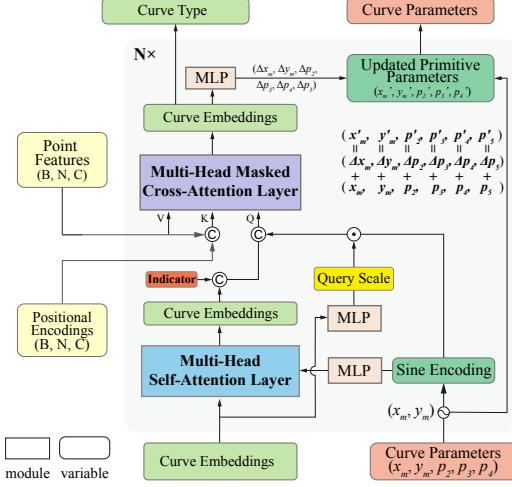


Figure 2: **The improved Transformer decoder.** The curve parameters are formulated as the position queries, while the curve embeddings encode the curve type. The curve parameters are updated in each Transformer layer through the predicted δ .

Methods	DeepCAD							Fusion 360 Gallery						
	Acc_t^{SKH}	Acc_p^{SKH}	CD	ECD	NC	IR	$\#\Delta P$	Acc_t^{SKH}	Acc_p^{SKH}	CD	ECD	NC	IR	$\#\Delta P$
<i>Methods of Representing Sketch with Implicit SDF Encoding</i>														
ExtrudeNet	34.58%	31.71%	0.379	0.962	0.849	24.19%	34.34	37.81%	34.39%	0.671	0.808	0.809	23.83%	27.47
Point2Cyl	41.37%	39.41%	0.489	1.027	0.819	3.91%	26.18	42.98%	41.16%	0.529	0.769	0.769	3.87%	26.96
SECAD-Net	45.91%	41.37%	0.341	0.868	0.861	8.03%	32.78	46.82%	42.97%	0.449	0.684	0.813	7.82%	28.61
<i>Generation Methods Based on Language Model</i>														
DeepCAD	82.61%	73.36%	0.898	1.883	0.823	14.12%	5.89	77.31%	69.81%	7.128	8.729	0.719	13.18%	7.47
HNC-CAD	84.31%	76.71%	0.827	1.064	0.846	6.01%	4.43	80.62%	72.19%	4.381	5.571	0.748	5.92%	6.14
<i>Methods Based on Primitive Fitting</i>														
UCSG-Net	-	-	1.849	1.174	0.820	-	14.19	-	-	0.952	1.277	0.770	-	11.17
CSG-Stump	-	-	3.031	0.755	0.828	-	19.46	-	-	0.781	0.991	0.744	-	13.08
Ours	96.14%	86.81%	0.312	0.581	0.897	3.71%	4.14	94.17%	83.52%	0.392	0.571	0.839	3.62%	5.17

Table 1: **Evaluation results** on the test set of the CAD reconstruction dataset. CAD reconstruction and generation methods are selected for comparison.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUPS

Datasets. To evaluate the proposed model, we conduct experiments on DeepCAD (Wu et al., 2021) and Fusion 360 Gallery (Willis et al., 2021). The training data is generated as described in Sec. 3.2.

Implementation Details. We use PointNet++ (Qi et al., 2017) for extrusion segmentation. The Point2Primitive is trained for 400 epochs with a total batch size of 64 and learning rate (lr) $1e-5$ with linear warmup and step reduce lr scheduler. As for the query denoising settings, the noise rate for the curve type and parameters is 0.2 and 0.3, respectively, with 5 denoising groups. The transformer encoder and decoder contain 6 and 8 transformer layers, respectively, with eight attention heads and latent model dimension 256.

Evaluation Metrics. We adopt command type accuracy (Acc_t^{SKH}) and parameter accuracy (Acc_p^{SKH}) for quantitative evaluations of the sketch prediction following (González-Lluch et al., 2021). For networks that represent sketch by the implicit neural field, the tools for convert sketch SDF to command are developed based on the tool by SECAD (Li et al., 2023a). Furthermore, following Chen and Zhang (2019), we also report chamfer distance (CD), edge chamfer distance (ECD), normal consistency (NC), and the number of generated primitives ($\#\Delta P$) to measure the quality of the recovered 3D geometry, more details of the quantitative metrics are shown in the Appendix B.

4.2 MAIN RESULTS

To demonstrate its effectiveness, we compare the proposed model to multiple existing methods, covering the methods of representing sketch with SDF encoding (Li et al., 2023a; Ren et al., 2022; Uy et al., 2022), the generation methods based on language model (Wu et al., 2021; Xu et al., 2023), and methods based on primitive fitting (Kania et al., 2020; Ren et al., 2021). The direct reconstruction results are visualized for comparison.

Table 1 summarizes the main results, and we can find that sketch curve type and parameter accuracy of the Point2Primitive are much higher than the other methods while the CD metric of the Point2Primitive is lower. The smallest CD and ECD value indicates that the presented CAD reconstruction method achieves better geometry fidelity while preserving accurate and sharp shape edges. This shows that by directly recovering curves of the extrusion primitive, our method can achieve better performance than the other extrusion-segmentation methods that represent the sketch with an implicit field. As for the number of generated primitives, our method is closer to human designs (The smallest $\#\Delta P$ value). This improvement is due to the explicit fine-grained reconstruction from each extrusion primitive; every parameter of the curves and extrusion operations are all predicted and optimized.

4.3 VISUALIZATION RESULTS

Figure 3 demonstrates the visualized reconstruction results on the DeepCAD and Fusion 360 Gallery datasets, respectively. It can be seen that the results of our method are more compact and complete, while the edges are sharp and sketches are accurate. This shows that our method can produce CAD reconstruction of high geometrical fidelity. The methods based on the language model can also

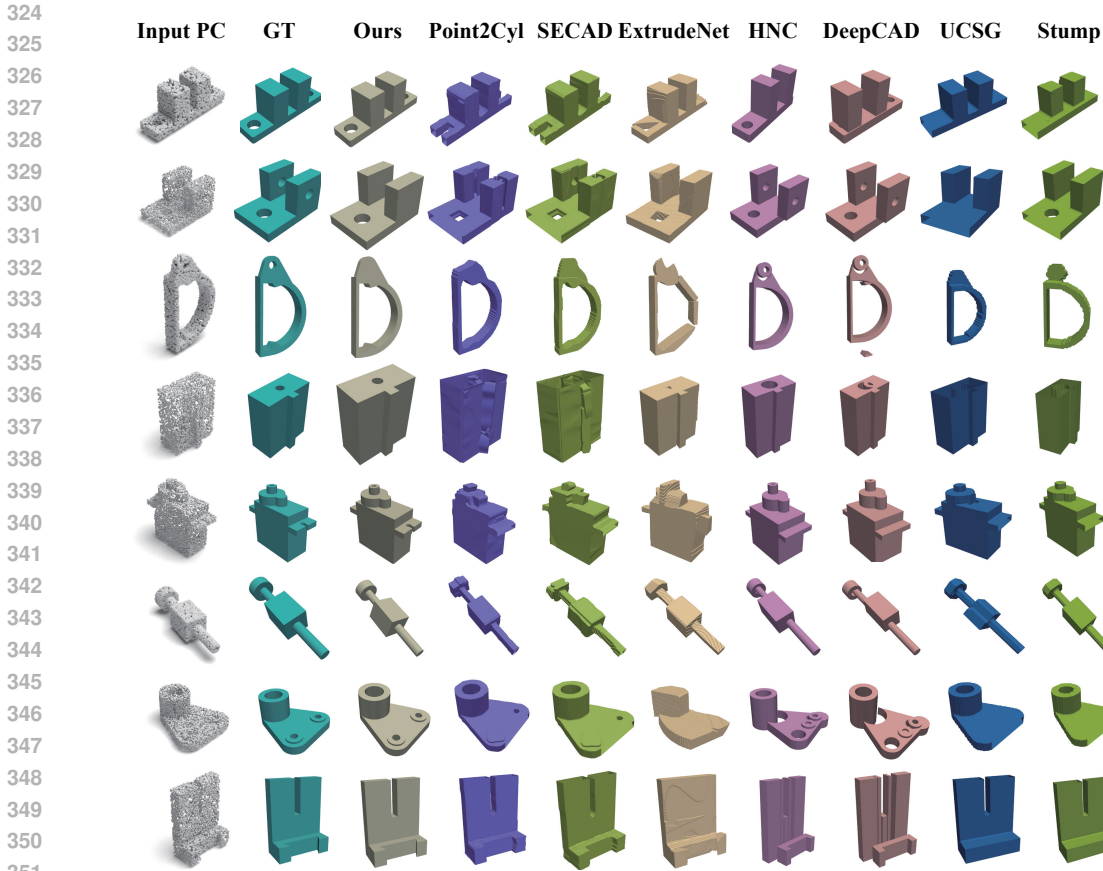


Figure 3: **Visual comparison** between reconstruction results on the DeepCAD (*the first 4 lines*) and Fusion360 gallery dataset (*the last 4 lines*).

Methods	Acc_t^{SKH}	Acc_p^{SKH}	CD	ECD	NC	$\#\Delta P$
<i>Methods of Representing Sketch with Implicit SDF Encoding</i>						
ExtrudeNet	28.39%	25.61%	0.614	1.117	0.776	36.14
Point2Cyl	34.47%	30.25%	0.518	1.065	0.791	27.96
SECAD-Net	36.61%	31.43%	0.437	1.079	0.806	34.18
<i>Generation Method Based on Language Model</i>						
DeepCAD	61.41%	43.71%	5.919	6.883	0.708	7.16
HNC-CAD	63.39%	53.29%	6.864	7.064	0.711	6.56
<i>Methods Based on Primitive Fitting</i>						
UCSG-Net	-	-	2.146	1.273	0.797	14.81
CSG-Stump	-	-	3.681	0.958	0.804	20.16
Ours	94.67%	83.81%	0.410	0.607	0.819	4.97

Table 2: **Evaluation results** on the Augmented DeepCAD dataset.

produce accurate geometry structure, but the error of the parameters in the sequence results in some models of poor compactness. The primitive detection methods (USCG and Stump) fail to predict some holes in the model. Still, the edges are critically more accurate than the ExtrudeNet, whose reconstructions are less complete. Please refer to Appendix I for more visualizations.

4.4 COMPARISON ON AUGMENTED DEEPCAD DATASET

In addition to the experiments on the original DeepCAD and Fusion 360 Gallery datasets, we conducted extra comparisons on the augmented DeepCAD datasets. More specifically, we add random noise to the parameters of the CAD modeling sequence while keeping the types intact. This noise causes the augmented DeepCAD test set to have model structures similar to the original shape but with some modifications to the geometry details. We train each method on the original DeepCAD dataset and test all the methods on the augmented DeepCAD test sets.

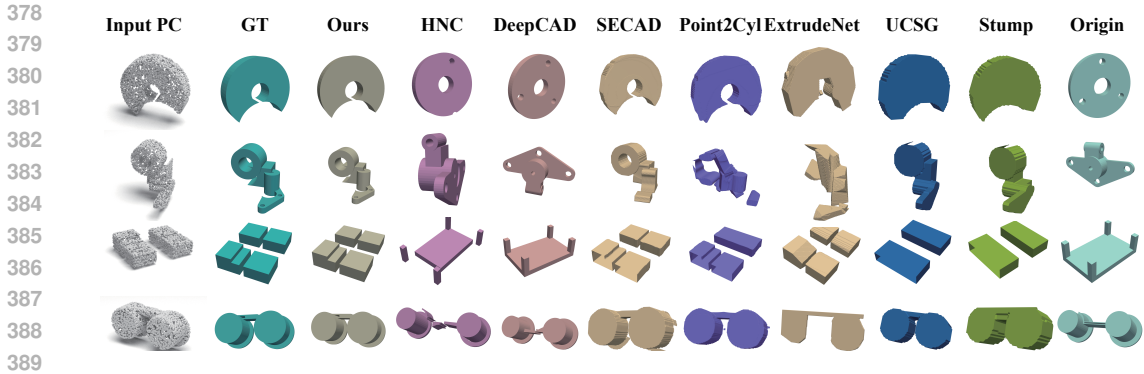
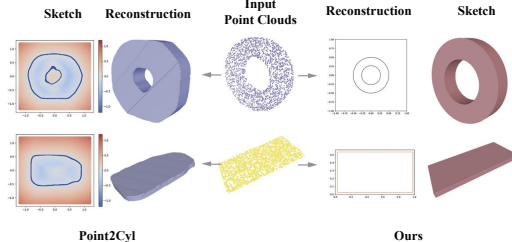


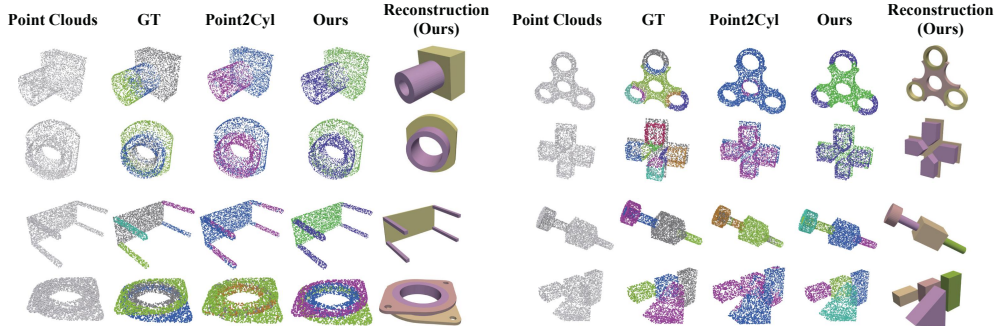
Figure 4: **Visual comparison** between reconstruction results on the augmented DeepCAD dataset.

Metrics	Ours		Point2Cyl	
	w/o \mathcal{L}_{skh}	w/	w/o \mathcal{L}_{skh}	w/
Ext. Seg IoU	0.814	0.862	0.801	0.817
BB. Seg Acc	0.902	0.902	0.867	0.867
EA. Angle Err ↓	7.416	7.173	8.391	8.267
Fit Ext. ↓	0.0771	0.0527	0.0791	0.0741

(a) Quantitative comparison on DeepCAD.



(b) There constructed sketches by Point2Cyl and Point2Primitive.



(c) The extrusion segmentation by Point2Cyl and Point2Primitive.

Figure 5: **Comparison with Point2Cyl.**

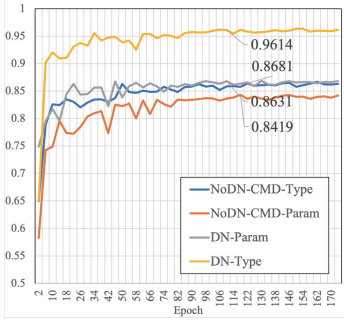
Table 2 demonstrates the quantitative results of the augmented DeepCAD dataset. We add the original shape (Origin) to the last column in the visualized results as shown in Figure 4. It can be seen that the methods based on the language model see a direct drop in the metrics of the sketch command type and command parameter, leading to a significant increase in the CD metric. Some reconstruction results by the language model preserve similarities to the original shape, leading to false reconstruction. Please refer to Appendix I for more visualizations.

The generated results are similar to the ground truth but with low geometry accuracies. On the other hand, reconstruction methods can still produce results that satisfy geometry fidelity. This implies that the methods based on the language model are less generalizable than the other methods. Also, both the quantitative metrics and the visual results verify that the proposed method can be generalized to reconstruction tasks beyond the train-test dataset.

4.5 DETAILED COMPARISON WITH POINT2CYL

In Table 5a, we present the quantitative comparison between our method and the most related approach, Point2Cyl. Compared with Point2Cyl, the metrics of the extrusion (Ext.) and barrel/based (BB.) segmentation are improved by 5.6% and 3.5%, respectively, leading to 13% and 32% reduc-

432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485



(a) Accuracy curve.

No.	β	SkhRep	ImpDec	Acc_t^{SKH}	Acc_p^{SKH}	CD
0	2.0	-	-	82.19%	72.49%	0.819
1	2.0	✓	-	85.71%	78.61%	0.674
2	1.0	✓	✓	93.16%	84.91%	0.341
3	3.0	✓	✓	93.24%	84.73%	0.337
Ours	2.0	✓	✓	96.14%	86.81%	0.312

(b) Quantitative results of ablation study.

Figure 6: Ablation study.

tion on the extrusion axis (EA.) angle and fitting (Fit Ext.) error, respectively. This shows that by supervising the extrusion segmentation with curve loss (type and parameter loss), more geometrical information is introduced, leading to better geometry accuracy.

We visualized the sketch predicted by Point2Cyl and our method, as shown in Figure 5b. It can be seen that because Point2Cyl represents the sketch by SDF, the edges of the sketch profiles are curved, while the counterpart of Point2Primitive is sharp and accurate. In addition, we present more visualizations of the predicted sketch by Point2Primitive in the Appendix C. Figure 5c (b) demonstrates the segmentation results by the two methods. It can be seen that even though the multiple parts of one extrusion are labeled with different extrusion labels, the segmentation network can still learn the geometry representation and cluster the points as the same extrusion. This implies that our method can predict simplified primitives that are closer to human design.

4.6 ABLATION STUDY

We perform ablation studies to carefully analyze the query denoising (DN), the improved transformer decoder, the sketch primitive definition, and the balancing weight in the loss function. All quantitative metrics are measured on the DeepCAD dataset.

Query denoising. It can be seen from the convergence curve shown in Figure 6a that by employing denoising in the training procedure, the ultimate sketch prediction accuracy is improved by 11.4% and 3.1% on Acc_t^{SKH} and Acc_p^{SKH} , respectively. Please refer to Appendix D for more visualizations. From the curves, one can see that by directly predicting the curve parameters and types, the accuracy is extremely high (type error below 0.5% and parameter error below 0.06).

Decoder and Sketch Hierarchy. We replace the improved decoder layer (ImpDec) with the vanilla transformer. Also, we utilize the sketch representation (SkhRep) method in Wu et al. (2021), which converts parameter prediction into a classification problem. As shown in Figure 6b, the Acc_t^{SKH} improves by 12.2% by utilizing the ImpDec. Also, the center-prior sketch hierarchy contributes to an 8.4% improvement in Acc_p . This shows the effectiveness of the proposed improved transformer decoder.

Balancing Weight. Figure 6a shows that the balance weight $\beta = 2$ performs slightly better than the other conditions, and we set it to our default value.

5 CONCLUSION

The proposed Point2Primitive can produce CAD reconstruction of high geometric fidelity. The reconstruction is more accurate using curves as the sketch representation instead of SDF. The curve parameters are accurate by direct prediction in an autoregressive way through the improved transformer decoder. In further work, we plan to extend our approach to include more complex modeling commands.

REFERENCES

- 486
487
488 Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *2019 IEEE/CVF*
489 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6
- 490 Carmen González-Lluch, Raquel Plumed, David Pérez-López, Pedro Company, Manuel Contero, and Jorge D.
491 Camba. A constraint redundancy elimination strategy to improve design reuse in parametric modeling.
492 *Computers in Industry*, 129:103460, 2021. 6
- 493 Kacper Kania, Maciej Zięba, and Tomasz Kajdanowicz. Ucsq-net – unsupervised discovering of constructive
494 solid geometry tree, 2020. 2, 6
- 495
496 Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. 3
- 497 Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc
498 Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning, 2019.
499 1
- 500 JP Kruth and A Kerstens. Reverse engineering modelling of free-form surfaces from point clouds subject to
501 boundary conditions. *JOURNAL OF MATERIALS PROCESSING TECHNOLOGY*, 76(1-3):120–127, 1998.
502 13th International Conference on Computer-Aided Production Engineering, WARSAW, POLAND, JUN 11-
503 13, 1997. 1
- 504 Eric-Tuan Le, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and NiloyJ. Mitra. Cpfm:
505 Cascaded primitive fitting networks for high-resolution point clouds. *arXiv: Computer Vision and Pattern*
506 *Recognition, arXiv: Computer Vision and Pattern Recognition*, 2021. 2
- 507
508 Feng Li, Hao Zhang, Shilong Liu, Jian Guo, LionelM. Ni, and Lei Zhang. Dn-detr: Accelerate detr training by
509 introducing query denoising. 5
- 510 Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J. Guibas. Supervised fitting of ge-
511 ometric primitives to 3d point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern*
512 *Recognition (CVPR)*, 2019. 2
- 513 Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-ming Yan. Secad-net: Self-supervised cad reconstruction by
514 learning sketch-extrude operations. 2023a. 1, 2, 6
- 515
516 Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on
517 \mathcal{X} -transformed points, 2018. 16
- 518 Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. Surface and edge detection for
519 primitive fitting of point clouds, 2023b. 2
- 520
521 Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection.
522 In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 5
- 523 Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. Point2cad: Reverse engineering cad
524 models from 3d point clouds, 2023. 2
- 525
526 Weijian Ma, Shuaiqi Chen, Yunzhong Lou, Xueyang Li, and Xiangdong Zhou. Draw step by step: Reconstruct-
527 ing cad construction sequences from point clouds via multimodal diffusion. In *CVPR*, pages 27144–27153,
528 2024. 3
- 529 Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deep sdf: Learning
530 continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer*
531 *Vision and Pattern Recognition (CVPR)*, 2019. 2
- 532
533 Thomas Paviot. pythonocc, 2022. 4
- 534 CharlesR. Qi, Li Yi, Hao Su, and LeonidasJ. Guibas. Pointnet++: Deep hierarchical feature learning on point
535 sets in a metric space. *Cornell University - arXiv, Cornell University - arXiv*, 2017. 4, 6
- 536 Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse
537 sketch-and-extrude for shape parsing. pages 482–498. Springer Nature Switzerland. 2
- 538
539 Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan,
Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. Csg-stump: A learning friendly csg-like representation for
interpretable shape parsing, 2021. 2, 6

- 540 Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse
541 sketch-and-extrude for shape parsing. 2022. 1, 6
- 542 Gopal Sharma, Difan Liu, Subhansu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch.
543 Parsenet: A parametric surface fitting network for 3d point clouds. *arXiv: Computer Vision and Pattern
544 Recognition, arXiv: Computer Vision and Pattern Recognition*, 2020. 2
- 545 Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph Lambourne, Tolga Birdal, and
546 Leonidas Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In
547 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11840–11850,
548 2022. 1, 6
- 549 Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf-: Neural radiance fields
550 without known camera parameters, 2022. 2
- 551 Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama,
552 and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction
553 from human design sequences, 2021. 1, 3, 6
- 554 Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design
555 models. *Cornell University - arXiv, Cornell University - arXiv*, 2021. 3, 6, 9
- 556 Xiang Xu, Karl D. D. Willis, Joseph G. Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka
557 Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks.
558 2022. 3
- 559 Xiang Xu, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Karl D. D. Willis, and Yasutaka Furukawa.
560 Hierarchical neural coding for controllable cad model generation, 2023. 3, 6
- 561 Fan Yang, Sicheng Zhao, Yanhao Zhang, Haoxiang Chen, Hui Chen, Wenbo Tang, Haonan Lu, Pengfei Xu,
562 Zhenyu Yang, Jungong Han, and Guiguang Ding. Llmi3d: Empowering llm with 3d perception from a single
563 2d image, 2024. 3
- 564 Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang.
565 Capri-net: Learning compact cad shapes with adaptive primitive assembly, 2021. 3
- 566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593

594 A THE USE OF LARGE LANGUAGE MODELS (LLMs)

595
596 In this work, Large Language Models (LLMs) were used solely to polish the language for clarity
597 and readability. No LLMs were employed for idea generation, experimental design, data analysis,
598 or any other part of the research process.

600 B EVALUATION METRICS

602 B.1 PRIMITIVE TYPE AND PARAMETER METRICS

603
604 The command type and command parameter accuracy evaluate how accurate the predicted command
605 is. For each CAD command, we calculate the command type accuracy by:

$$606 \quad ACC_{\text{type}} = \frac{1}{N_c} \sum_{i=1}^{N_c} \Psi[t_i = \hat{t}_i], \quad (9)$$

609 where N_c denote the total number of the CAD commands, C_i and \hat{C}_i are the ground truth com-
610 mand type and predicted command type, respectively. $\Psi[i] \in \{0, 1\}$ is the indicator function. The
611 command parameter accuracy is calculated by:

$$612 \quad ACC_{\text{param}} = \frac{1}{K} \sum_{i=1}^{N_c} \sum_{j=1}^{|\hat{p}_i|} \Phi[|p_{i,j} - \hat{p}_{i,j}| < \epsilon] \Psi[C_i = \hat{C}_i], \quad (10)$$

615 where $K = \sum_{i=1}^{N_c} \Psi[C_i = \hat{C}_i] |\hat{p}_i|$ is the total number of parameters in all correctly recovered
616 commands. $p_{i,j}$ and $\hat{p}_{i,j}$ are ground-truth command parameters and predicted command parameters.
617 ϵ is the tolerance threshold accounting for the parameter accuracy. In practice, we use $\epsilon = 0.01$.

619 B.2 EXTRUSION SEGMENTATION IOU

620
621 To evaluate the extrusion segmentation, we use the Segmentation (Seg) IoU as the metric. The
622 predicted extrusion segmentation labels are first reordered to correspond with the GT through Hun-
623 garian matches. The Seg IoU can be formulated as follows:

$$624 \quad \text{Seg IoU} = \frac{1}{K} \sum_{k=1}^K RIoU(\mathbb{1}(\hat{\mathbf{W}}_{:,k}), \mathbf{W}_{:,k}), \quad (11)$$

$$627 \quad RIoU(\mathbf{X}, \mathbf{Y}) = \frac{\mathbf{X}^\top \cdot \mathbf{Y}}{\|\mathbf{X}\|_1 + \|\mathbf{Y}\|_1 - \mathbf{X}^\top \cdot \mathbf{Y}}, \quad (12)$$

629 where $\hat{\mathbf{W}}_{:,k}$ and $\mathbf{W}_{:,k}$ are the predicted and ground-truth labels of the k -th extrusion, respectively.
630 $\mathbb{1}(\cdot)$ indicates the one-hot conversion.

632 B.3 BASE/BARREL POINT SEGMENTATION

633
634 The Base/Barrel (BB) point Segmentation accuracy is defined as follows:

$$635 \quad \text{BB Acc} = \frac{1}{N} \sum_{i=1}^N (\mathbb{1}(\hat{\mathbf{B}}_{i,:}) == \mathbf{B}_{i,:}), \quad (13)$$

638 where $\hat{\mathbf{B}}_{i,:}$ and $\mathbf{B}_{i,:}$ are the predicted and ground-truth barrel label, respectively. N is the number of
639 input point clouds.

641 B.4 EXTRUSION AXIS ANGLE ERROR

642
643 Extrusion axis (EA) angle error measures the angle error between the GT and prediction, which is
644 defined as follows:

$$645 \quad \text{EA Angle Error} = \frac{1}{K} \sum_{k=1}^K \arccos(\hat{\mathbf{e}}_{k,:}^\top \mathbf{e}_{k,:}), \quad (14)$$

647 where $\hat{\mathbf{e}}_{k,:}^\top$ and $\mathbf{e}_{k,:}$ are the predicted and ground-truth extrusion axis, respectively.

B.5 EXTRUSION FITTING ERROR

The extrusion fitting (Fit Ext) error measures the average fitting error of each extrusion, which can be defined as follows:

$$\text{Fit Ext} = \frac{1}{K} \sum_{k=1}^K \mathcal{F}, \quad \text{where} \quad (15)$$

$$\mathcal{F} \triangleq \frac{1}{N_{\mathbf{P}_k}} \sum \left| \text{SDF}(\hat{\mathbf{s}}_k, \hat{\mathbf{S}}_k) - \text{SDF}(\mathbf{s}_k, \mathbf{S}_k) \right|, \quad \text{and} \quad \hat{\mathbf{S}}_k = \prod (\mathbf{P}^{\text{barrel}_k}, \hat{e}_{k,:}, \hat{c}_{k,:}), \quad (16)$$

where $\hat{c}_{k,:}$ is the predicted extrusion center. $\prod(\cdot)$ projects per extrusion barrel points using the extrusion axis and center. The inner summation $\mathcal{F}(\mathbf{P}, k)$ represents the goodness of the k -th extrusion fitting.

B.6 PRIMITIVE NUMBER

The primitive number is a metric of reconstructed sequence fidelity. We calculate the $\Delta\#P$ to measure the length difference between the reconstructed and ground-truth sequence, which is defined as follows:

$$\Delta\#P = \left| \hat{N}_c - N_c \right|, \quad (17)$$

where \hat{N}_c and N_c are the predicted and ground-truth primitive numbers.

C VISUALIZED SKETCH PREDICTION

We provide more visualized results of the predicted sketch. Fig. 7 demonstrates some examples of the predicted sketch primitives. It can be seen that the predicted sketches are accurate both in curve type and parameter. However, some parameter errors can be seen in the Figure. Therefore, we develop a post-optimized method to eliminate the parameter errors, which is detailed in Section F.2.

D VALIDATION LOSS W/O QUERY DENOISING

Except for the command type and accuracy curve, Fig. 8 demonstrates the validation loss of the proposed Point2Primitive during training. From the loss curves, we can see that the final loss of the curve type trained with DN is much smaller than the counterpart without DN (0.413 vs 1.1809). As for the curve parameter, the L-1 loss drops by 3.7%, and the distance loss drops by 4.9%. Also, the convergence time trained with DN is shorter than without DN.

E THE IMPLEMENTATION OF THE EXTRUSION SEGMENTATION NETWORK

E.1 EXTRUSION AND BARREL POINT SEGMENTATION

Figure 9 demonstrates the extrusion segmentation network. PointNet++ is used as the encoder and the decoder. The Normal head and Ext head produce the normal prediction \hat{N} and the extrusion segmentation logits $\hat{\mathbf{M}}$. The extrusion segmentation logits $\hat{\mathbf{M}}$ produce the $2K$ -classes of each point. The $(2k)$ -th elements of each row of the $\hat{\mathbf{M}}$ encode the extrusion label, while the $(2k + 1)$ -th elements encode the barrel label. Thus the extrusion $\hat{\mathbf{W}}$ and barrel $\hat{\mathbf{B}}$ segmentation logits are formulated as follows:

$$\hat{\mathbf{W}}_{:,j} = \hat{\mathbf{M}}_{:,2j} + \hat{\mathbf{M}}_{:,2j+1}, \quad (18)$$

$$\hat{\mathbf{B}}_{:,0} = \sum_j \hat{\mathbf{M}}_{:,2j}, \quad \text{and} \quad \hat{\mathbf{B}}_{:,1} = \sum_j \hat{\mathbf{M}}_{:,2j+1}. \quad (19)$$

702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755

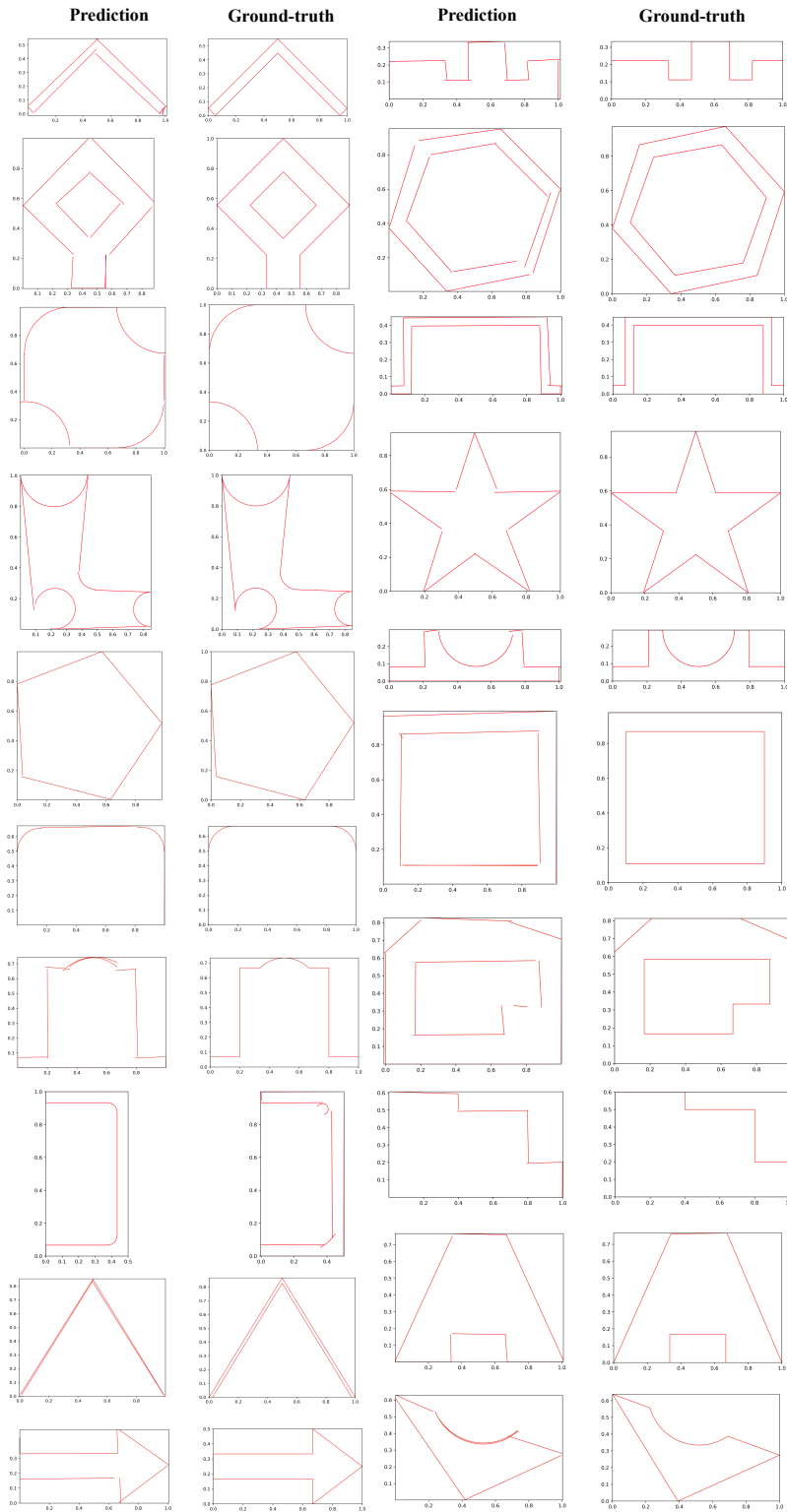


Figure 7: Visualization of the sketch primitive prediction.

756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809

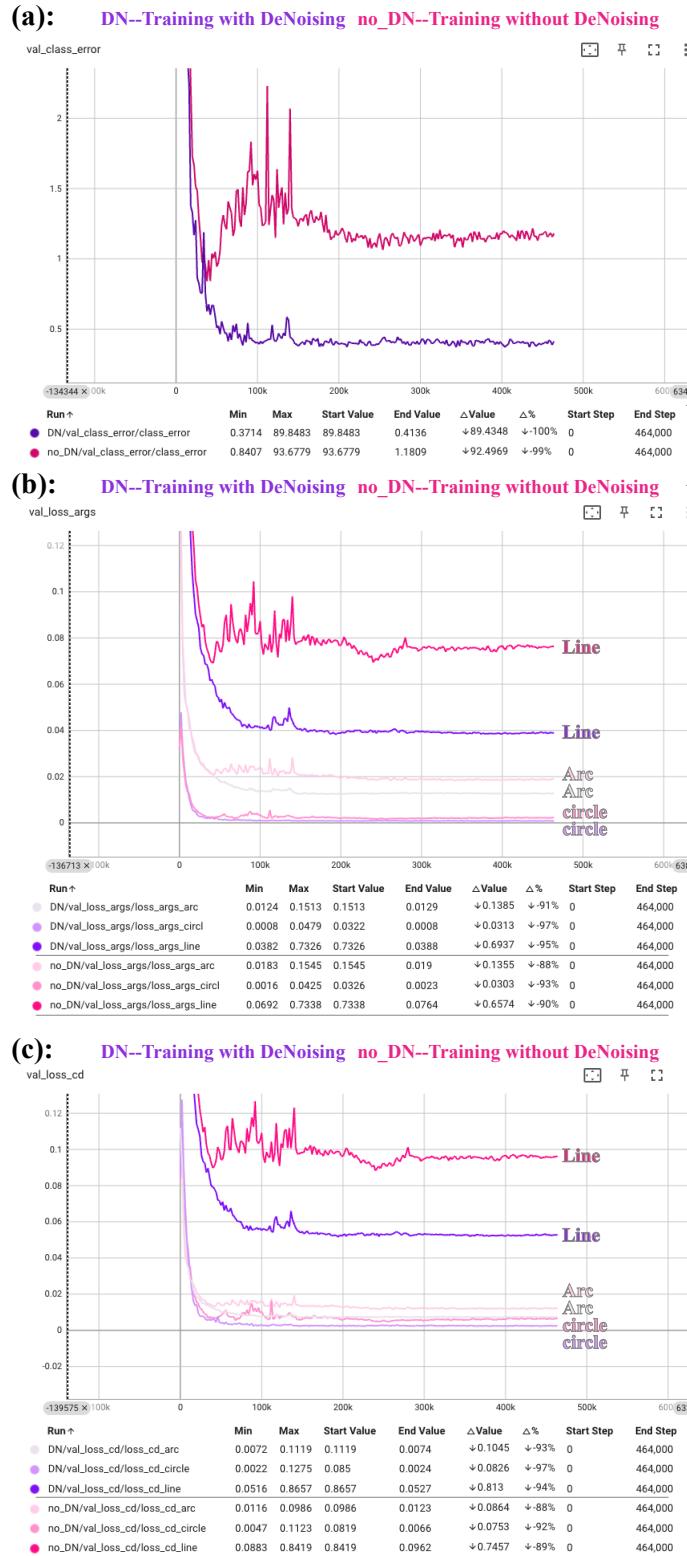


Figure 8: Validation curves with and without query denoising of the Point2Primitive during training; (a) the class error during training; (b) the L-1 loss of the primitive parameter during training; (c) the distance loss of the primitive parameter during training.

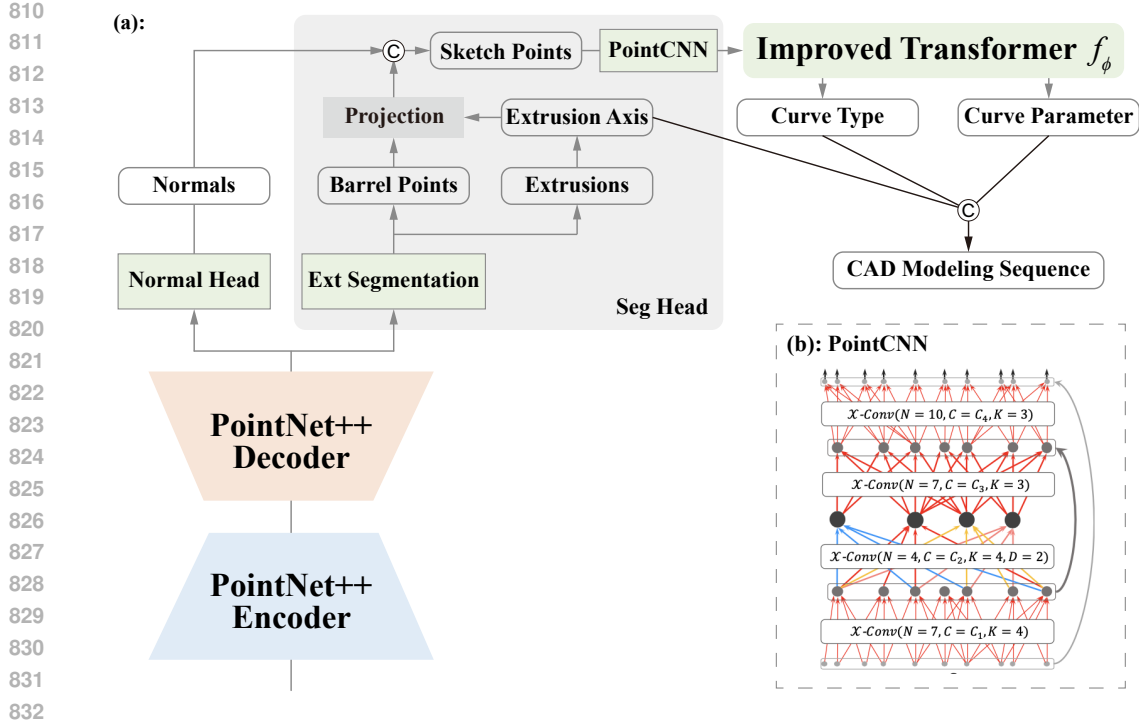


Figure 9: The implementation of the extrusion segmentation network: (a) the network architecture of the Point2Primitive; (b) the Sketch Point Encoder based on the PointCNN

E.2 SKETCH POINT ENCODER

As shown in Figure 9, the sketch point encoder is developed based on the PointCNN(Li et al., 2018). Instead of using random or fps points, fixed points are set as the representation points. Also, to provide position encodings for the transformer, a learned position encoding is utilized.

F POST OPTIMIZATION

F.1 THE BATCH SDF COMPUTING

We develop a batch SDF computing method to calculate the sketch SDF considering batch input. More specifically, the curves of each sketch will be divided into three curve types (L, A, C), and curves of the same type are calculated in batches. We now provide an anonymous GitHub repository to present the SDF computing in https://github.com/AnonymousRepo1234/Point2Primitive/blob/main/SDF_batch_cal.py.

F.2 CURVE FINE-TUNING

To produce more accurate curve parameters for CAD reconstruction, we develop a post-optimization method for the predicted curve parameters, as shown in Figure 10. The predicted sketch points can be used to optimize the predicted curve parameters. First, we project the barrel point into sketch points. Then, we calculate the sketch SDF to the sketch points using batch SDF computing. The sketch SDF to its sketch points is set as the loss function. Thirdly, the curve parameters are set as model parameters and updated using the autograd in Pytorch supervised by the SDF. The sketch SDF to its sketch points is supposed to be zero. The algorithm is shown in Algorithm 1.

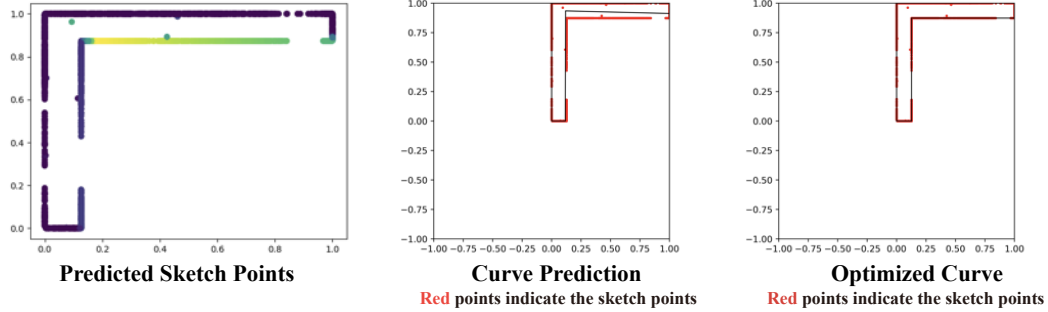


Figure 10: Post optimization of curve prediction.

Algorithm 1 Curve Fine-tuning

```

892 Input:  $Curves$ 
893 Output:  $T$  as modified Curves
894  $Curves_{norm} \leftarrow normalize(Curves)$ 
895  $Curves_{norm} \leftarrow sort(Curves_{norm})$  ▷ sorted by starting point
896  $S \leftarrow Curves_{norm}$ 
897  $T \leftarrow \emptyset$  ▷ as modified Curves
898 while  $S \neq \emptyset$  do
899    $x \leftarrow S.front$ 
900   for each curve  $l \in S$  do ▷ modify neighborhood line
901     if  $\|x.end, l.start\| < Threshold$  then
902        $x.end, l.start \leftarrow mid(x.end, l.start)$ 
903        $T \leftarrow T \cup \{x\}$ 
904        $S \leftarrow S \setminus \{x\}$ 
905       break
906     else if  $\|x.end, l.end\| < Threshold$  then
907        $x.end, l.end \leftarrow mid(x.end, l.end)$ 
908        $l \leftarrow reverse(l)$ 
909        $T \leftarrow T \cup \{x\}$ 
910        $S \leftarrow S \setminus \{x\}$ 
911       break
912     end if
913   end for
914 end while
915  $T \leftarrow post\_process(T)$ 

```

F.3 LOOP OPTIMIZATION

Algorithm 2 Loop Optimization

Input: point_cloud, Curves
Output: params as modified sketch curve params
 eval_points \leftarrow in_pc(point_cloud) ▷ as pc to be formulated
 cmd, params \leftarrow Initialize(sketch) ▷ cmd as mask of curve type, params as shape of sketch
for each epoch in *epoches* **do**
 line_loss \leftarrow SDF(eval_points, params[line_mask(cmd)]) ▷ calculate line, arc and circle loss
 arc_loss \leftarrow SDF(eval_points, params[arc_mask(cmd)])
 circle_loss \leftarrow SDF(eval_points, params[circle_mask(cmd)])
 loss \leftarrow concat(line_loss, arc_loss, circle_loss, -1)
 loss.min() ▷ minimize the last element of loss tuple
 loss.backward()
end for

The sketch should contain loops, which are composed of curves connected end to end. To connect the predicted curves into loops, we use an optimization method based on Greedy Algorithm as shown in Algorithm 2.

G EXTRUSION PARAMETERS CALCULATION

We calculate the parameters of the extrusion operation by the obtained barrel points. The parameters of the extrusion operation $E_i = (\hat{e}_i, \hat{t}_i)$ consists of the extrusion axis \hat{e}_i , extrusion position \hat{o}_i , and extrusion extent \hat{t}_i . Firstly, the optimal extrusion axis of an input extrusion cylinder PC is given by $\hat{e}_i = \operatorname{argmin}_{e_i, \|e_i\|=1} (e_i^T H_\phi e_i)$, where:

$$H_\phi = \mathbf{N}^T \Phi_{barr}^T \Phi_{barr} \mathbf{N} - \mathbf{N}^T \Phi_{base}^T \Phi_{base} \mathbf{N}, \quad (20)$$

where \mathbf{N} denotes the normals of the input points. $\Phi_{barr} = \operatorname{diag}(\phi_{barr})$, $\Phi_{base} = \operatorname{diag}(\phi_{base})$. ϕ_{base} and ϕ_{barr} indicate the barrel and base weights (assigned to all points) predicted by the segmentation head, respectively. The solution is given by the eigenvector corresponding to the smallest eigenvalue of H_ϕ .

The extrusion extent \hat{t} is the point projection range on the extrusion axis, which can be calculated by Equation 21.

$$\hat{t}_{min} = \min_{P_i \in P_{barr}} (e_i \cdot (P_i - \hat{o})) \quad \text{and} \quad \hat{t}_{max} = \max_{P_i \in P_{barr}} (e_i \cdot (P_i - \hat{o})), \quad (21)$$

H MORE DETAILS OF THE IMPROVED TRANSFORMER DECODER

H.1 COMMAND PARAMETER NOISE

For each input training pair (p_i, S_i) , we add random noises to both their curve types and curve parameters. A random noise $(\Delta x, \Delta y)$ is added to the endpoint coordinates. The values of $(\Delta x, \Delta y)$ are limited between $[0.0, 0.2]$ so that the noised endpoint will not shift too far from the ground-truth position. The other primitive parameters (x_{m_i}, y_{m_i}, r_i) are randomly sampled from $(\phi x_{m_i}, \phi y_{m_i}, \phi r_i)$, $\phi \in [-1, 1]$.

For curve type noising, the GT command types are randomly flipped.

H.2 ATTENTION MASK

An attention mask is used to prevent information leakage. There are two reasons for this. Firstly, the matching part might be able to obtain information from the noised curve types and easily infer the target curve types. Secondly, the various noised groups of the same ground-truth curve might exchange information with each other.

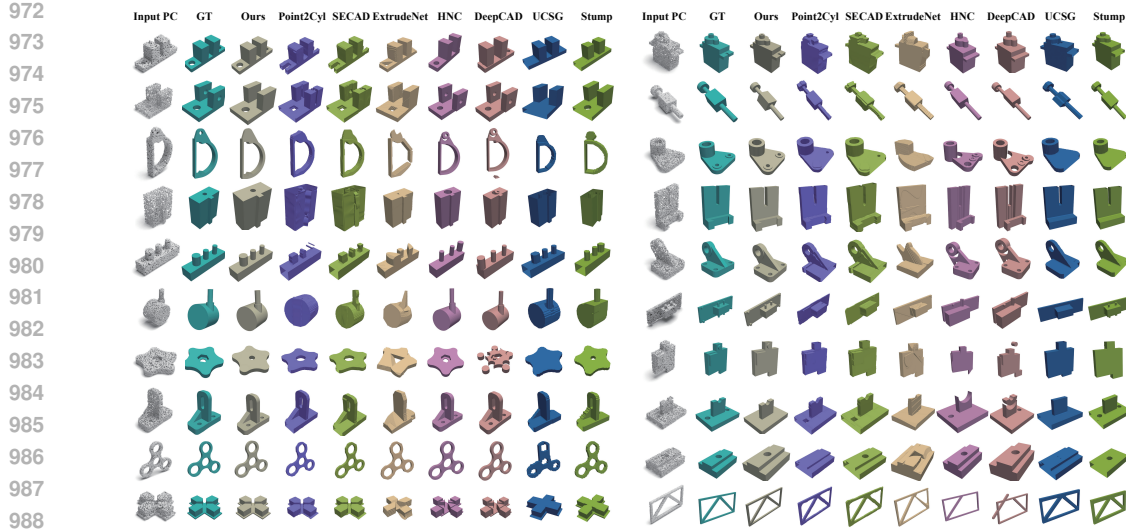


Figure 11: **More visual comparison** between reconstruction results on the DeepCAD and Fusion360 gallery dataset.

Given a sketch contains N_c curves. The noised ground-truth curves of all the primitives are first divided into K groups. The denoising part is then denoted as:

$$q = \{g_0, g_1, \dots, g_{K-1}\}, \quad \text{and} \quad g_j = \{q_0^k, q_1^k, \dots, q_{M-1}^k\} \quad (22)$$

where g_k is the k -th denoising group. q_m^k is the m -th query in the denoising group. Each denoising group contains M queries where M is the number of commands in one input PC.

The attention mask $A = [a_{ij}]_{W \times W}$ can then be formulated as:

$$a_{ij} = \begin{cases} 1, & \text{if } j < K \times M \text{ and } \lfloor \frac{i}{M} \rfloor \neq \lfloor \frac{j}{M} \rfloor; \\ 1, & \text{if } j < K \times M \text{ and } i \geq K \times M; \\ 0, & \text{else} \end{cases}, \quad (23)$$

where $a_{ij} = 1$ means the i -th query cannot see the j -th query and $a_{ij} = 0$ other wise.

The decoder embeddings are taken as curve embeddings in our model. Therefore, an indicator is appended to the curve embeddings to distinguish the denoising part from the matching part, as shown in Figure 2, where 1 means the query belongs to the denoising part and 0 means the query belongs to the matching part.

I MORE VISUALIZATIONS

We provide more visualization results as shown in Figure 11 and Figure 12.

J APPLICATIONS

In addition to the public CAD dataset, we also tested our method on some practical datasets. Figure 13 demonstrates some practical applications of the presented CAD reconstruction method.

1026
 1027
 1028
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079

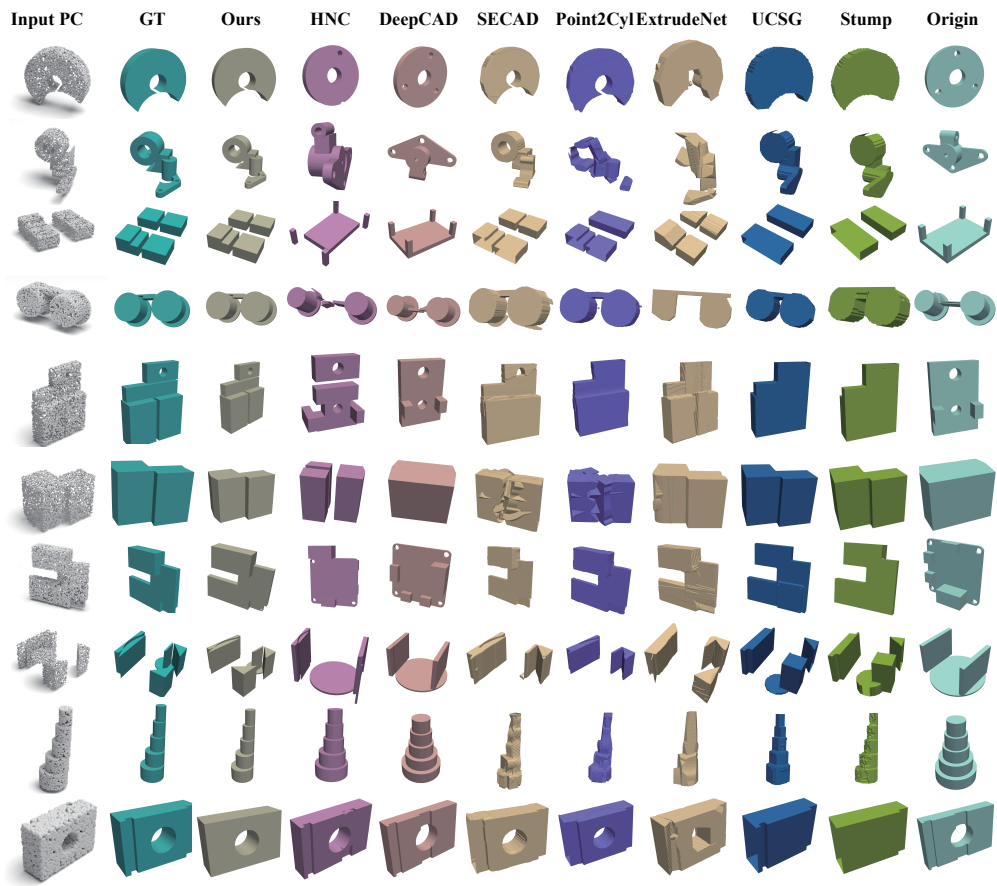


Figure 12: **More visual comparison** between reconstruction results on the augmented DeepCAD dataset.

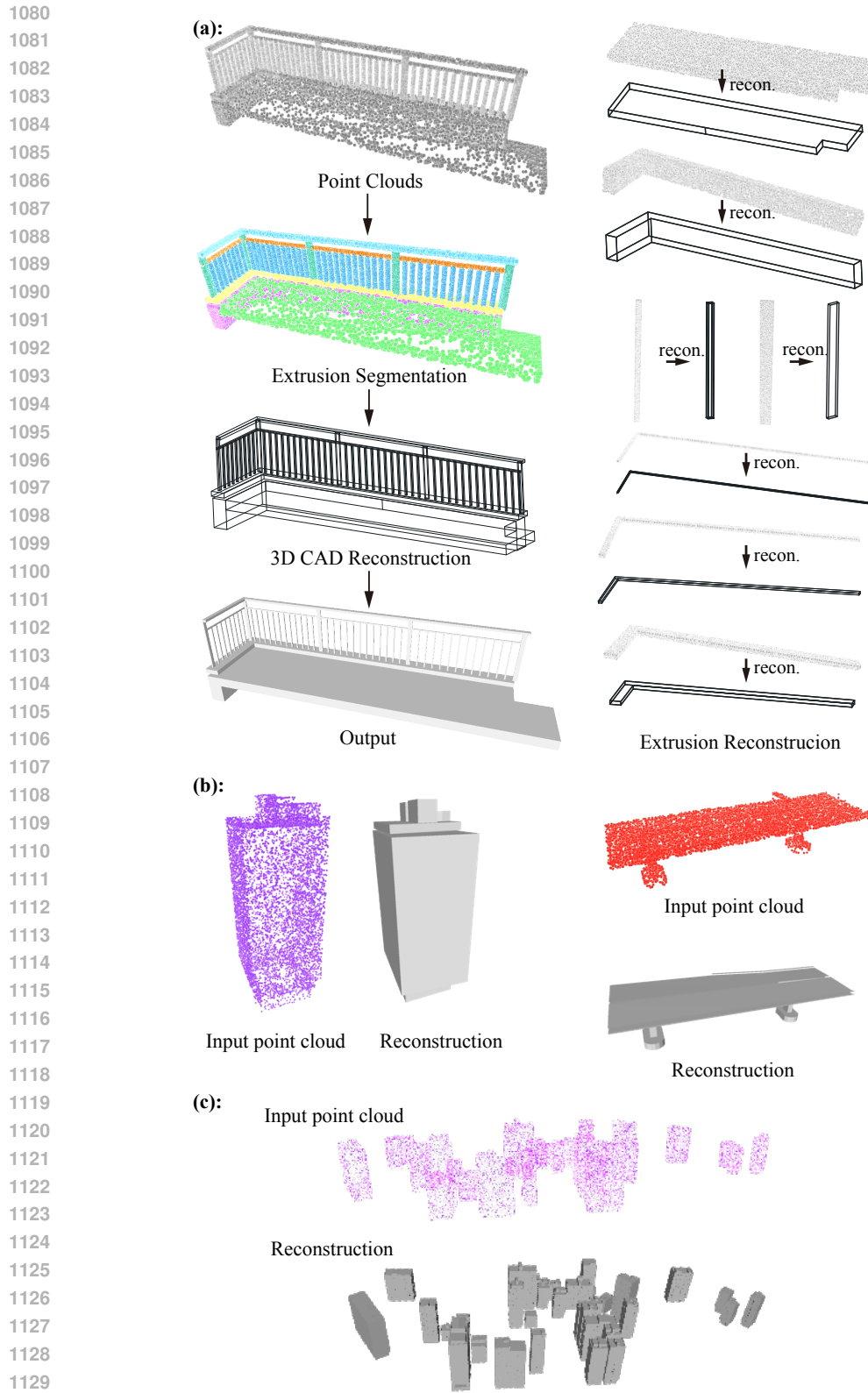


Figure 13: Examples of some practical application: (a) reconstruction of the structural components; (b) reconstruction of the buildings; (c) reconstruction of the small urban agglomeration.