
MiCADangelo: Fine-Grained Reconstruction of Constrained CAD Models from 3D Scans

Supplementary Material

1 Overview

2 This supplementary document provides additional technical details complementing the main paper,
3 along with an extended set of qualitative evaluations of the proposed MiCADangelo. Section 0.1
4 outlines the data preprocessing procedures, Section 0.2 describes the evaluation metrics, Section 0.3
5 provides additional experimental analysis, Section 0.4 discusses current limitations and failure cases,
6 and Section 0.5 presents additional qualitative results.

7 0.1 Data Preprocessing

8 This section describes the data preparation steps used for training and evaluation of the proposed
9 method.

10 0.1.1 Data Preparation for the Training of Plane Detection Network

11 **Preprocessing of extrusion planes.** To train the proposed plane detection network, extrusion planes
12 are extracted from the DeepCAD [1] dataset using the original train and test splits. For every extrusion
13 in a CAD sequence, we start from its sketch plane (\mathbf{o}, \mathbf{n}) with origin $\mathbf{o} \in \mathbb{R}^3$ and normal $\mathbf{n} \in \mathbb{R}^3$,
14 forward extent $e_1 \in \mathbb{R}$ and backward extent $e_2 \in \mathbb{R}$. We compute a single, canonically oriented plane
15 $(\mathbf{o}^*, \mathbf{n}^*)$. First, the origin \mathbf{o} is moved depending on the `extent_type` flag provided by the CAD
16 sequence:

$$\mathbf{o}^* = \begin{cases} \mathbf{o} - e_1 \mathbf{n}, & \text{if extent_type = symmetric extrusion ,} \\ \mathbf{o} - e_2 \mathbf{n}, & \text{if extent_type = two-sided extrusion ,} \\ \mathbf{o}, & \text{if extent_type = one-sided extrusion .} \end{cases}$$

17 The two limiting points are $\mathbf{o}_{\text{fwd}} = \mathbf{o} + e_1 \mathbf{n}$ and $\mathbf{o}_{\text{bwd}} = \mathbf{o} + e_2 \mathbf{n}$. The normalized connecting
18 vector $\mathbf{n}' = (\mathbf{o}_{\text{fwd}} - \mathbf{o}_{\text{bwd}}) / \|\mathbf{o}_{\text{fwd}} - \mathbf{o}_{\text{bwd}}\|$ is used as the normal direction. To make the direction
19 unambiguous, we flip it towards the positive axis $\mathbf{n}^* = |\mathbf{n}'|$. If the direction is flipped, \mathbf{o}^* is moved
20 to the point \mathbf{o}_{bwd} . Finally, the set $\{(\mathbf{o}_j^*, \mathbf{n}_j^*)\}_{j=1}^{n_s}$ where n_s is the number of extrusions, constitutes
21 the ground-truth extrusion planes.

22 **Preparation of Slicing Plane Labels.** For each CAD model in the DeepCAD dataset, we sample
23 equally spaced 40 slicing planes along each of the x -, y -, and z -axes, producing a total of $N = 120$
24 slicing planes:

$$\Pi = \{\pi_{a,\sigma} \mid a \in \{x, y, z\}, \sigma \in \{0, \dots, 39\}\}.$$

25 We define a binary label set $\mathbf{y} = \{y_i\}_{\pi_i \in \Pi}$, where $y_i \in \{0, 1\}$ for each slicing plane π_i , and initialize
26 all labels to 0. For every ground-truth plane π_k , we identify the nearest slicing plane:

$$\pi_k^* = \arg \min_{\pi_i \in \Pi} d(\pi_i, \pi_k),$$

27 and update the corresponding label y_k to 1, where $d(\pi_i, \pi_k)$ denotes the distance between planes π_i
28 and π_k .

29 0.1.2 Data Preparation for the Training of Sketch Parameterization Network

30 The constrained sketch parameterization network is initially trained on the SketchGraphs dataset [2].
31 It is then fine-tuned on an augmented version of SketchGraphs that includes added noise and
32 synthetically generated closed-loop images. This augmentation helps the model better adapt to the

characteristics of cross-sectional slice data. During training, a synthetic sketch is used if a uniformly sampled random number exceeds 0.5; otherwise, the original SketchGraphs sketch is retained. A random rotation between $[0, 2\pi]$ is also applied to the sketch with random probability 0.2. The procedures for generating random sketches and noise augmentation are detailed in Algorithm 1 and Algorithm 2, respectively. Figure 1 shows example sketches from the original SketchGraphs dataset and its augmented counterpart, highlighting the added variability introduced by synthetic loops and image-space noise.

Algorithm 1 GenerateRandomLoopSketch

Require: maxPrimitives, arcWeight

Ensure: $\mathcal{K} = (\mathcal{P}, \mathcal{C})$ ▷ Sketch primitives \mathcal{P} and constraints \mathcal{C}

```

1:  $\mathcal{P} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset$ 
2:  $n_p \leftarrow \text{random\_int}(3, \text{maxPrimitives})$ 
3:  $P \leftarrow \text{RandomPolygon}(n_p)$  ▷ Random polygon with at least 3 vertices
4: for  $i = 0$  to  $n_p - 1$  do
5:    $(\mathbf{x}_s, \mathbf{x}_e) \leftarrow \text{edge } i \text{ of } P$ 
6:   if  $\text{rand}() > \text{arcWeight}$  then
7:      $\mathbf{p}_i \leftarrow \text{Line}(\mathbf{x}_s, \mathbf{x}_e)$  ▷ Create line primitive
8:   else
9:      $\mathbf{x}_m \leftarrow \frac{1}{2}(\mathbf{x}_s + \mathbf{x}_e)$  ▷ Midpoint of the start and end points
10:     $\mathbf{v} \leftarrow \mathbf{x}_e - \mathbf{x}_s$ 
11:     $\delta \leftarrow \text{rand\_uniform}(0.05, 0.15)$ 
12:     $\mathbf{x}_a \leftarrow \mathbf{x}_m + \delta \cdot \mathbf{v}_\perp$  ▷ Add offset to midpoint
13:    if  $\text{rand}() > 0.8$  then
14:      swap  $\mathbf{x}_s \leftrightarrow \mathbf{x}_e$  ▷ Randomly reverse arc direction
15:    end if
16:     $\mathbf{p}_i \leftarrow \text{Arc}(\mathbf{x}_s, \mathbf{x}_a, \mathbf{x}_e)$  ▷ Create arc primitive
17:  end if
18:   $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{p}_i\}$  ▷ Add primitive to sketch
19:  if  $i > 0$  then
20:     $\mathbf{c}_i \leftarrow (\mathbf{p}_{i-1}, \mathbf{p}_i, \text{coincident})$  ▷ Add coincident constraint
21:     $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{c}_i\}$ 
22:  end if
23: end for
24:  $\mathbf{c}_{n_p} \leftarrow (\mathbf{p}_{n_p-1}, \mathbf{p}_0, \text{coincident})$  ▷ Close loop with final constraint
25:  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{c}_{n_p}\}$ 
26: return  $\mathcal{K} = (\mathcal{P}, \mathcal{C})$ 
```

0.1.3 Data Preparation for the Experimental Evaluation

DeepCAD Complex Subset. To assess the performance of the proposed method on more complex cases (Section 5.1 of the main paper), we selected a subset of CAD models from the DeepCAD test set based on the number of sketch loops. While a model can be complex even with a single loop, the DeepCAD dataset contains a disproportionately large number of very simple designs (e.g., cubes) that can dominate the overall evaluation. To mitigate this, we filtered for models containing more than four distinct loops. This process yielded a subset of 1,391 samples used for targeted evaluation.

Extruded SketchGraphs Dataset. To evaluate the impact of constraints in 3D reverse engineering (Section 5.2 of the main paper), we construct a set of solids with their corresponding constrained sketches from the Sketchgraphs [2] dataset. Starting from the first sketch in the test split, we iterate sequentially through the dataset and process only sketches that form closed loops. Each closed sketch is loaded into FreeCAD [3], placed at the origin with its normal aligned to $+z$, and extruded by 0.3 units along that axis to produce a 3D solid. The solid is then exported in three formats: OBJ, STEP, and native FCStd. This process is repeated until 1,000 solids are generated.

¹Implemented using polygenerator and shapely libraries.

Algorithm 2 RenderWithNoiseAugmentation

Require: Sketch \mathcal{K} , is_hand_drawn **Ensure:** Sketch image \mathbf{X}

```
1:  $s \leftarrow 128$ 
2: if  $\text{rand}() < 0.2$  then
3:    $s \leftarrow \text{rand\_choice}(\{64, 128, 256\})$   $\triangleright$  Random rescaling size
4: end if
5:  $\mathbf{X} \leftarrow \text{RenderSketch}(\mathcal{K}, \text{is\_hand\_drawn}, s)$   $\triangleright$  Render sketch at resolution  $s$ 
6:  $\mathbf{X} \leftarrow \text{Resize}(\mathbf{X}, 128 \times 128)$   $\triangleright$  Normalize size
7: if  $\text{rand}() < 0.2$  then
8:    $\mathbf{X} \leftarrow \text{AddNoiseNearForeground}(\mathbf{X})$   $\triangleright$  Add local noise
9: end if
10: if  $\text{rand}() < 0.2$  then
11:    $k \leftarrow \text{rand\_choice}(\{3, 5\})$ 
12:    $d \leftarrow 2k + 1$ 
13:    $\mathbf{X} \leftarrow \text{GaussianBlur}(\mathbf{X}, \text{kernel\_size} = d, \sigma = 0)$   $\triangleright$  Apply blur with random kernel size
14: end if
15: return  $\mathbf{X}$ 
```

Algorithm 3 AddNoiseNearForeground

Require: Image \mathbf{X} of size $H \times W$, max offset d **Ensure:** Modified image \mathbf{X}

```
1:  $B \leftarrow \{(x, y) \mid \mathbf{X}[y, x] = 0\}$   $\triangleright$  Foreground (black) pixel coordinates
2: if  $|B| = 0$  then
3:   return  $\mathbf{X}$ 
4: end if
5:  $N \leftarrow \text{randint}(0, 100)$   $\triangleright$  Number of noise points
6: for  $i = 1$  to  $N$  do
7:    $(x, y) \sim \text{Uniform}(B)$   $\triangleright$  Sample uniformly from foreground
8:    $\delta_x, \delta_y \leftarrow \text{randint}(-d, d)$ 
9:    $x' \leftarrow \text{clip}(x + \delta_x, 0, W - 1)$ 
10:   $y' \leftarrow \text{clip}(y + \delta_y, 0, H - 1)$   $\triangleright$  Clip to bounds
11:  if  $\text{rand}() < 0.5$  then
12:     $\mathbf{X}[y', x'] \leftarrow 0$   $\triangleright$  Add synthetic black pixel
13:  else
14:     $\mathbf{X}[y', x'] \leftarrow \text{randint}(0, 20)$   $\triangleright$  Add light gray speckle
15:  end if
16: end for
17: return  $\mathbf{X}$ 
```

54 **Constrained Sketch Transformations.** As discussed in Section 5.2 of the main paper, the impact
55 of geometric constraints is evaluated by applying controlled transformations to the sketches in the
56 *Extruded SketchGraphs* dataset. For each sketch, we randomly displace a single point, triggering
57 a constraint-driven update to the overall geometry. This results in a modified sketch and a corre-
58 sponding transformed 3D shape. In total, this procedure yields 1,000 CAD models with associated
59 transformations. To enable applying the same transformation to predicted sketches, we record the
60 original point and its displacement vector during transformation generation. In the predicted sketch,
61 we identify the closest point—among the start, midpoint, or end of all primitives—to the originally
62 displaced point, and apply the same displacement vector. Examples of such transformations are
63 shown in Figure 5 of the main paper and Figure 2 in the supplementary.

64 **DeepCAD Cross-section Images.** To evaluate the performance of the sketch parameterization
65 network on cross-sectional data (Section 5.3 of the main paper), a total of 5,106 cross-section images
66 are extracted from single-extrusion models in the DeepCAD test set. Each cross-section image is
67 generated by slicing the corresponding 3D mesh along the sketch plane specified in the associated
68 CAD sequence.

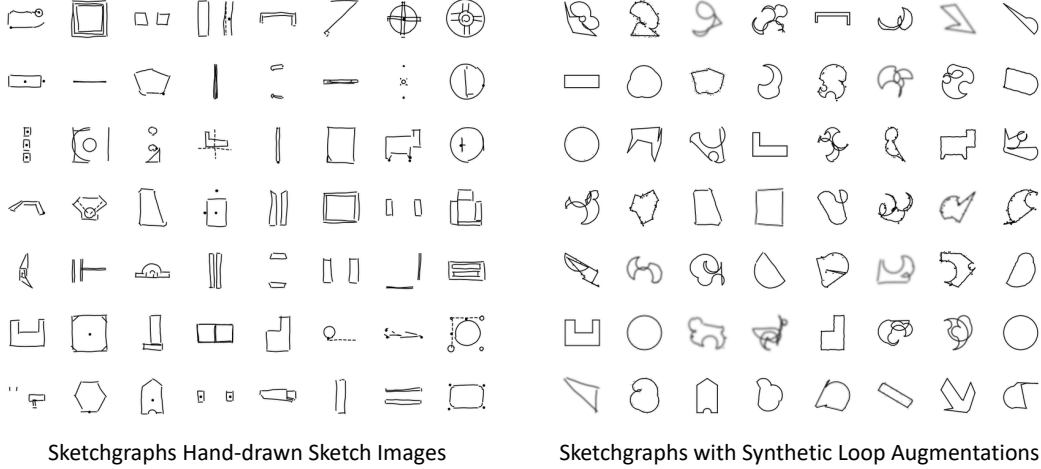


Figure 1: Examples of original (left) and augmented SketchGraphs sketches (right), illustrating the effect of synthetic loop generation and noisy rendering.

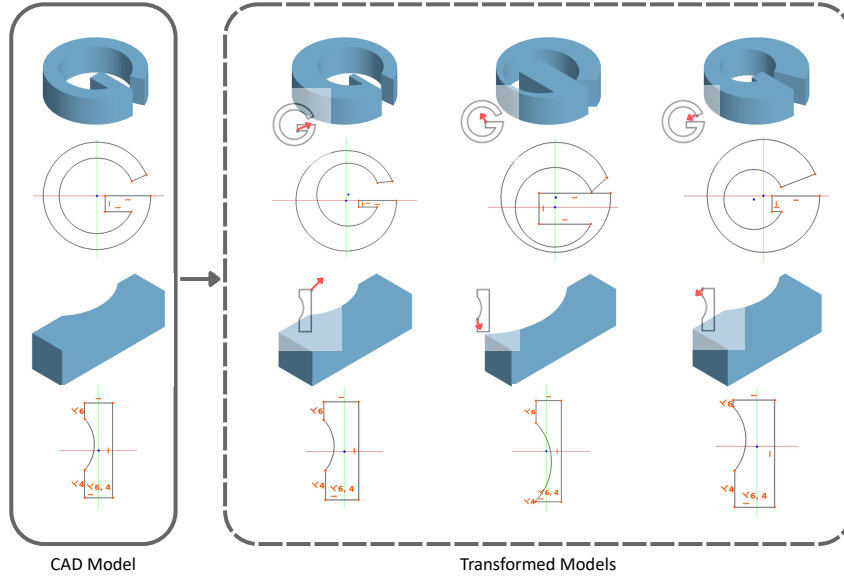


Figure 2: Examples of constrained sketch transformations and their effect on the resulting 3D models.

69 0.2 Metrics

70 Quantitative evaluation of the proposed method was conducted using the metrics described below.

71 **Chamfer Distance (CD).** Let M_{pred} and M_{gt} denote the meshes obtained from the predicted and
 72 ground-truth CAD models, respectively. A fixed number of points are uniformly sampled from the
 73 surface of each mesh, yielding two point sets:

$$\mathbb{P} = \{\mathbf{p}_i\}_{i=1}^N, \quad \hat{\mathbb{P}} = \{\hat{\mathbf{p}}_i\}_{i=1}^N,$$

74 where $N = 8192$ and $\mathbf{p}_i, \hat{\mathbf{p}}_i \in \mathbb{R}^3$. Sampled points \mathbb{P} and $\hat{\mathbb{P}}$ are normalized to unit scale. The
 75 *Chamfer Distance* between the sampled point sets is then defined as:

$$\text{CD} = \frac{1}{2N} \sum_{i=1}^N \min_{\hat{\mathbf{p}}_j \in \hat{\mathbb{P}}} \|\mathbf{p}_i - \hat{\mathbf{p}}_j\|_2^2 + \frac{1}{2N} \sum_{j=1}^N \min_{\mathbf{p}_i \in \mathbb{P}} \|\hat{\mathbf{p}}_j - \mathbf{p}_i\|_2^2. \quad (1)$$

76 **Intersection over Union (IoU).** To evaluate the volumetric similarity between two 3D meshes, we
 77 compute the *Intersection over Union (IoU)*. Let M_{pred} and M_{gt} be the predicted and ground-truth

78 meshes obtained from corresponding CAD models, each possibly composed of multiple connected
 79 components $\mathbf{M}_{\text{pred}} = \bigcup_i \mathbf{M}_{\text{pred}}^{(i)}$, $\mathbf{M}_{\text{gt}} = \bigcup_j \mathbf{M}_{\text{gt}}^{(j)}$. The *IoU* is defined as

$$\text{IoU}(\mathbf{M}_{\text{pred}}, \mathbf{M}_{\text{gt}}) = \frac{\sum_{i,j} \text{Vol}(\mathbf{M}_{\text{pred}}^{(i)} \cap \mathbf{M}_{\text{gt}}^{(j)})}{\sum_i \text{Vol}(\mathbf{M}_{\text{pred}}^{(i)}) + \sum_j \text{Vol}(\mathbf{M}_{\text{gt}}^{(j)}) - \sum_{i,j} \text{Vol}(\mathbf{M}_{\text{pred}}^{(i)} \cap \mathbf{M}_{\text{gt}}^{(j)})} \quad (2)$$

80 where $\text{Vol}(\cdot)$ denotes the volume of a mesh².

81 **Edge Chamfer Distance (ECD).** To evaluate performance on boundary curves, we compare models
 82 using points sampled only from their edges. For each STEP file obtained from prediction and ground-
 83 truth, we draw $M = 4096$ points uniformly across all edge curves, center the point cloud at the
 84 origin, and normalize it to the unit bounding box, yielding two sets

$$\mathbb{B} = \{\mathbf{b}_i\}_{i=1}^M, \quad \hat{\mathbb{B}} = \{\hat{\mathbf{b}}_i\}_{i=1}^M,$$

85 where $\mathbf{b}_i, \hat{\mathbf{b}}_i \in \mathbb{R}^3$. The *Edge Chamfer Distance* between the predicted and ground-truth edge sets is
 86 then defined as:

$$\text{ECD} = \frac{1}{2M} \sum_{i=1}^M \min_{\mathbf{b}_j \in \mathbb{B}} \|\hat{\mathbf{b}}_i - \mathbf{b}_j\|_2^2 + \frac{1}{2M} \sum_{i=1}^M \min_{\hat{\mathbf{b}}_j \in \hat{\mathbb{B}}} \|\mathbf{b}_i - \hat{\mathbf{b}}_j\|_2^2. \quad (3)$$

87 **Invalidity Ratio (IR).** The *Invalidity Ratio (IR)* is defined as the percentage of predicted CAD models
 88 that fail to produce valid meshes during export.

89 **Sketch Chamfer Distance (SCD).** To assess the similarity between predicted and ground-truth
 90 sketches, we compute a 2-D Chamfer Distance between the sets of foreground pixel coordinates
 91 obtained from their $h \times w$ binary rasterizations. Let the sets of foreground pixel coordinates be

$$\mathbb{S} = \{\mathbf{s}_n\}_{n=1}^{N_f}, \quad \hat{\mathbb{S}} = \{\hat{\mathbf{s}}_n\}_{n=1}^{\hat{N}_f}$$

92 where $\mathbf{s}_n, \hat{\mathbf{s}}_n \in \{(i, j) \mid 1 \leq i \leq h, 1 \leq j \leq w\}$, and N_f, \hat{N}_f denote the number of foreground
 93 pixels in the ground-truth and predicted rasterizations, respectively. The bidirectional *Sketch Chamfer*
 94 *Distance* is then defined as

$$\text{SCD} = \frac{1}{2\hat{N}_f} \sum_{n=1}^{\hat{N}_f} \min_{\mathbf{s}_k \in \mathbb{S}} \|\hat{\mathbf{s}}_n - \mathbf{s}_k\|_2^2 + \frac{1}{2N_f} \sum_{n=1}^{N_f} \min_{\hat{\mathbf{s}}_k \in \hat{\mathbb{S}}} \|\mathbf{s}_n - \hat{\mathbf{s}}_k\|_2^2. \quad (4)$$

95 **Plane Detection Metrics (Precision, Recall, F1).** To evaluate the performance of the sketch plane
 96 detection network, we treat each slicing plane across all CAD models as a binary classification
 97 instance. Let $y_i \in \{0, 1\}$ be the ground-truth label of the i -th slicing plane ($y_i = 1$ if the plane
 98 coincides with a sketch plane, 0 otherwise) and $\hat{y}_i \in \{0, 1\}$ the corresponding network prediction.
 99 Then, *Precision*, *Recall* and *F1* scores are computed as follows:

$$\text{TP} = \sum_i \mathbb{1}[y_i = 1 \wedge \hat{y}_i = 1], \quad \text{FP} = \sum_i \mathbb{1}[y_i = 0 \wedge \hat{y}_i = 1], \quad \text{FN} = \sum_i \mathbb{1}[y_i = 1 \wedge \hat{y}_i = 0].$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{F1} = \frac{2 \text{TP}}{2 \text{TP} + \text{FP} + \text{FN}} = \frac{2 \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

²The volume of a mesh is computed using trimesh library.

0.3 Additional Experimental Analysis

This section extends the main paper’s experimental analysis with further evaluations on complex geometries, plane detection, error accumulation, and cut extrusions.

0.3.1 Complex Geometries

We further extend the complex geometry analysis by evaluating a subset of highly complex models containing more than eight loops (271 models). These models represent parts with multiple intricate geometric details. The quantitative comparisons on these partitions are reported in Table 1.

Method	Mean CD↓	Median CD↓	IR↓	IoU↑	ECD↓
CADSIG-Net [4]	6.48	1.64	5.1	41.07	4.84
Ours	5.47	0.45	5.5	64.30	2.12

Table 1: Performance on models containing more than 8 loops.

0.3.2 Plane Detection

In Table 4 of the main paper, we ablate the impact of contextual embeddings on plane detection performance. We extend this analysis to evaluate their effect on the full CAD reconstruction pipeline. The results are presented in the Table 2. We observe that contextual embeddings have an effect on the overall CAD reconstruction performance on all reported metrics. Nonetheless, it is interesting to observe that, despite significantly underperforming in cross-section plane detection, our method without contextual embeddings still achieves reasonable overall CAD reconstruction performance. This can be attributed to the robustness of the subsequent stages—sketch parameterization and extrusion optimization—which are capable of recovering plausible reverse engineering design paths, even when the detected cross-section planes deviate from the ground truth.

Contextual Emb.	Mean CD↓	Median CD↓	IR↓	IoU↑	ECD↓
✗	8.80	0.39	3.5	69.4	2.25
✓	2.27	0.20	2.6	80.6	0.46

Table 2: Comparison of CAD reconstruction with and without contextual embeddings in plane detection.

The value of N controls the density of cross-section slice sampling, determining the granularity of the input to the reconstruction pipeline. We conduct an ablation on DeepCAD dataset with different values in Table 3. We observe that larger values provide the input model with a more dense input representation and slightly increase CAD reconstruction performance across metrics. In this work, we set N to 40, which offers a good balance between accuracy and computational cost. A similar compromise arises in point-cloud pipelines (as in all competitors [4, 5, 6, 1]), where the input cloud is routinely down-sampled to a fixed number of points.

N	Mean CD↓	Median CD↓	IR↓	IoU↑	ECD↓
10	3.20	0.26	4.5	78.2	0.54
20	3.17	0.25	4.3	78.4	0.53
40	2.27	0.20	2.6	80.6	0.46

Table 3: Ablation on the number of 2D cross-sectional slices N used as input to the slice plane detection network.

0.3.3 Error Accumulation Analysis

Error accumulation is an inherent challenge in all recent CAD reverse engineering approaches including MiCADangelo. This is due to the fact that a CAD model is typically represented as a sequence that is generated over a number of steps. Recent top-down approaches [4, 5] generate a CAD sequence autoregressively, meaning that inaccurately predicted sequence tokens are fed back into the model, potentially compounding errors and degrading the accuracy of subsequent token

predictions. Similarly, bottom-up approaches [6, 7, 8] involve multiple sequential stages, such as per-point segmentation, base or barrel label prediction, and extrusion parameter estimation, with error propagating at each stage. Nevertheless, to assess the robustness of the different steps of MiCADangelo, we evaluate its core stages independently (plane detection, sketch parameterization, and extrusion parameter optimization) in Table 4. In each case, we observe improved performance compared to the corresponding stages in related methods, which explains the superior overall performance achieved by our approach.

Plane Detection. We compare MiCADangelo’s plane detection performance against CAD-SIGNet [15], using standard precision, recall, and F1-score. Note that CADSIG-Net is designed to align plane detection with design history, hence its predicted and ground-truth planes are anchored in the original DeepCAD design planes. In contrast, MiCADangelo targets reverse engineering and thus operates on cross-sectional planes obtained by pre-processing DeepCAD (detailed in Supp. Section 0.1). The Table 4 shows that MiCADangelo achieves strong performance in cross-sectional plane detection, with notably higher metrics than CAD-SIGNet’s performance in the design-plane detection setting.

Sketch Parameterization. We extend our evaluation on Table 5 of the main paper with the addition of an autoregressive baseline, Vitruvion [4]. We show in the Table 4 that MiCADangelo outperforms both Vitruvion [4] and Davinci [35] in cross-section parameterization in terms of Sketch Chamfer Distance (SCD).

Extrusion Performance. To evaluate extrusion performance, we compare MiCADangelo with Point2Cyl [6], a bottom-up method that estimates extrusion cylinders from 3D point clouds. We use 100 CAD models from the dataset provided in [8] and, for each model, match each ground-truth extrusion cylinder to its closest predicted counterpart from both methods using Chamfer Distance (CD). We then report the average CD across all matched pairs. As shown in the Table 4, MiCADangelo achieves a lower average CD, indicating more accurate extrusion recovery. It is worth noting that, due to the high computational cost of Point2Cyl (over 4 hours for 100 samples), this evaluation was limited to 100 models.

Plane Detection				Extrusion Performance		Sketch Parameterization	
Method	Precision	Recall	F1	Method	Extrusion CD↓	Method	SCD↓
CADSIG-Net [4]	0.701	0.696	0.686	Point2Cyl [6]	27.9	Vitruvion [9]	1.236
Ours	0.894	0.864	0.870	Ours	10.1	Davinci [10]	0.827
						Ours	0.283

Table 4: Error accumulation analysis across plane detection, sketch parameterization, and extrusion performance.

0.3.4 Cut Extrusions

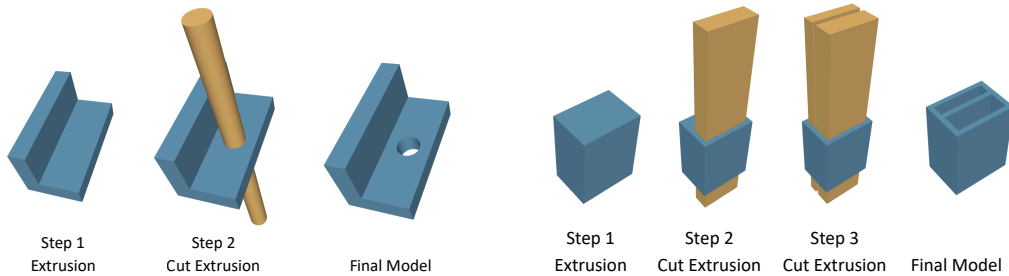


Figure 3: Example CAD reconstructions with cut extrusions.

Cut extrusions can be categorized into: (1) visible cut extrusions, when their originating sketch loops are visible and nested within other loops; (2) invisible cut extrusions, when their sketch loops are invisible in the final CAD geometry hence not captured by cross-sections. Our method handles visible cut extrusions as detailed in Section 4.4. The invisible cut extrusions are not directly handled. In such cases, the models are reverse engineered using alternative sketch-extrude sequences. Table 5 demonstrates that our method outperforms CAD-SIGNet in reconstructing models with visible and

invisible cut extrusions. Visual illustrations of individual reconstruction steps with cut extrusions are shown in Figure 3.

Method	Mean CD↓	Median CD↓	IR↓	IoU↑	ECD↓
CAD-SIGNet [4]	4.95	0.77	3.0	65.5	3.97
Ours	4.26	0.64	3.5	71.8	3.13

Table 5: Performance on DeepCAD models with cut extrusions.

0.4 Limitations and Failure Case Analysis

The proposed method is currently limited to extrusion-based CAD operations, consistent with prior work [4, 7, 6]. While extrusions are foundational to CAD modeling, many real-world designs rely on additional features such as revolutions, sweeps, lofts, and fillets. In the proposed method, extrusions are defined relative to sketch plane normals, which can lead to inaccurate outputs when the input geometry involves non-axis-aligned extrusions. Additionally, the proposed method does not support advanced sketch primitives such as B-splines, limiting its applicability to freeform designs. Future work will address these limitations by expanding the set of supported CAD operations, optimizing direction vectors in the differentiable extrusion process, and extending the sketch parameterization network to handle B-spline primitives.

The proposed method also exhibits some imperfections within its supported scope. We conduct a qualitative analysis of representative failure cases, as illustrated in Figure 4. Common issues include: (i) suboptimal extrusions, where the predicted extrusion height deviates from the ground truth; (ii) primitive simplifications, in which sketch elements such as arcs are approximated using coarse line segments, and (iii) missing or incorrectly predicted sketch planes, resulting in misaligned or absent features in the final 3D reconstruction. These examples highlight the difficulty of robustly predicting sketch planes, constrained sketches, and extrusion parameters, especially in models with complex geometry and topology.

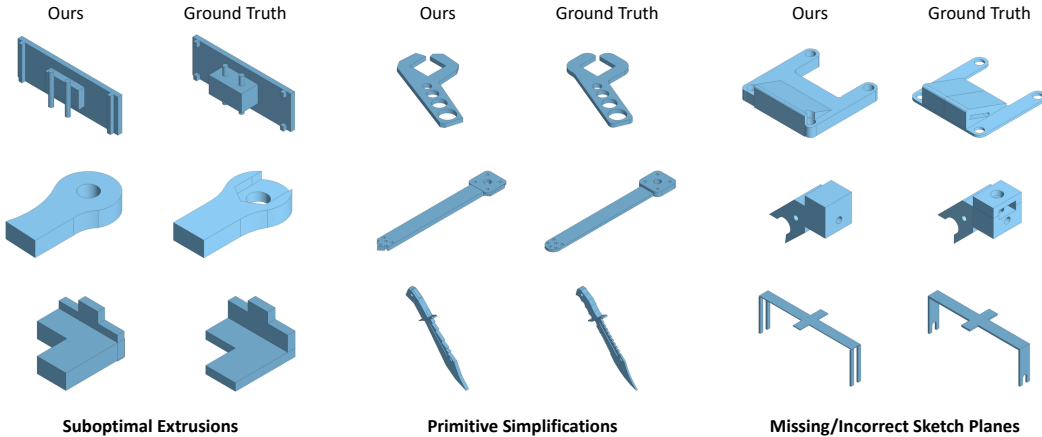


Figure 4: Representative failure cases illustrating issues such as inaccurate extrusion height, primitive simplification, and sketch plane misprediction.

183 0.5 Additional Qualitative Results

184 This section provides additional qualitative results and visualizations. Figure 5 shows intermediate
 185 steps of reconstructed CAD models in FreeCAD [3]. Figure 6 shows the visualization of differentiable
 186 extrusion. Figure 7, Figure 8, Figure 9, and Figure 10 shows more qualitative comparisons on
 187 DeepCAD and Fusion360 [11] datasets.

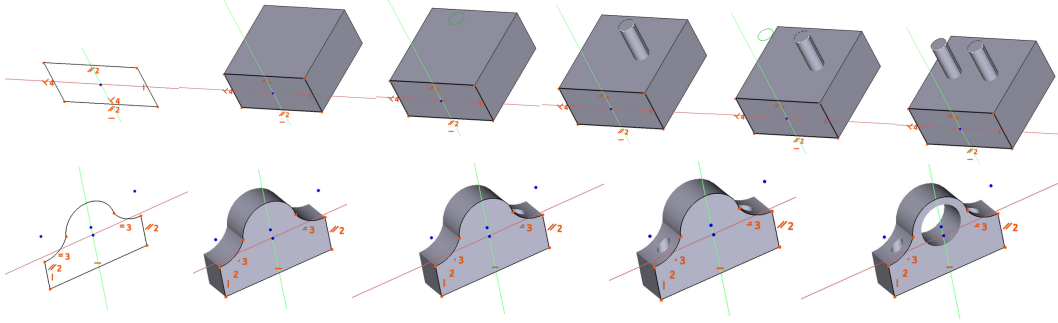


Figure 5: Visualization of intermediate steps in constrained CAD reconstructions in FreeCAD.

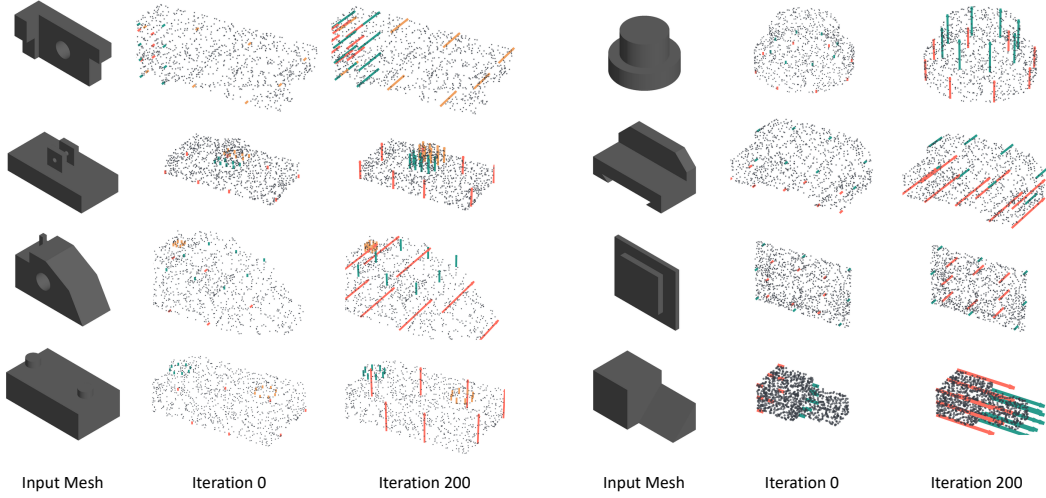


Figure 6: Visualizations of extrusion optimization with the points sampled from the input mesh and the extrusion vectors.

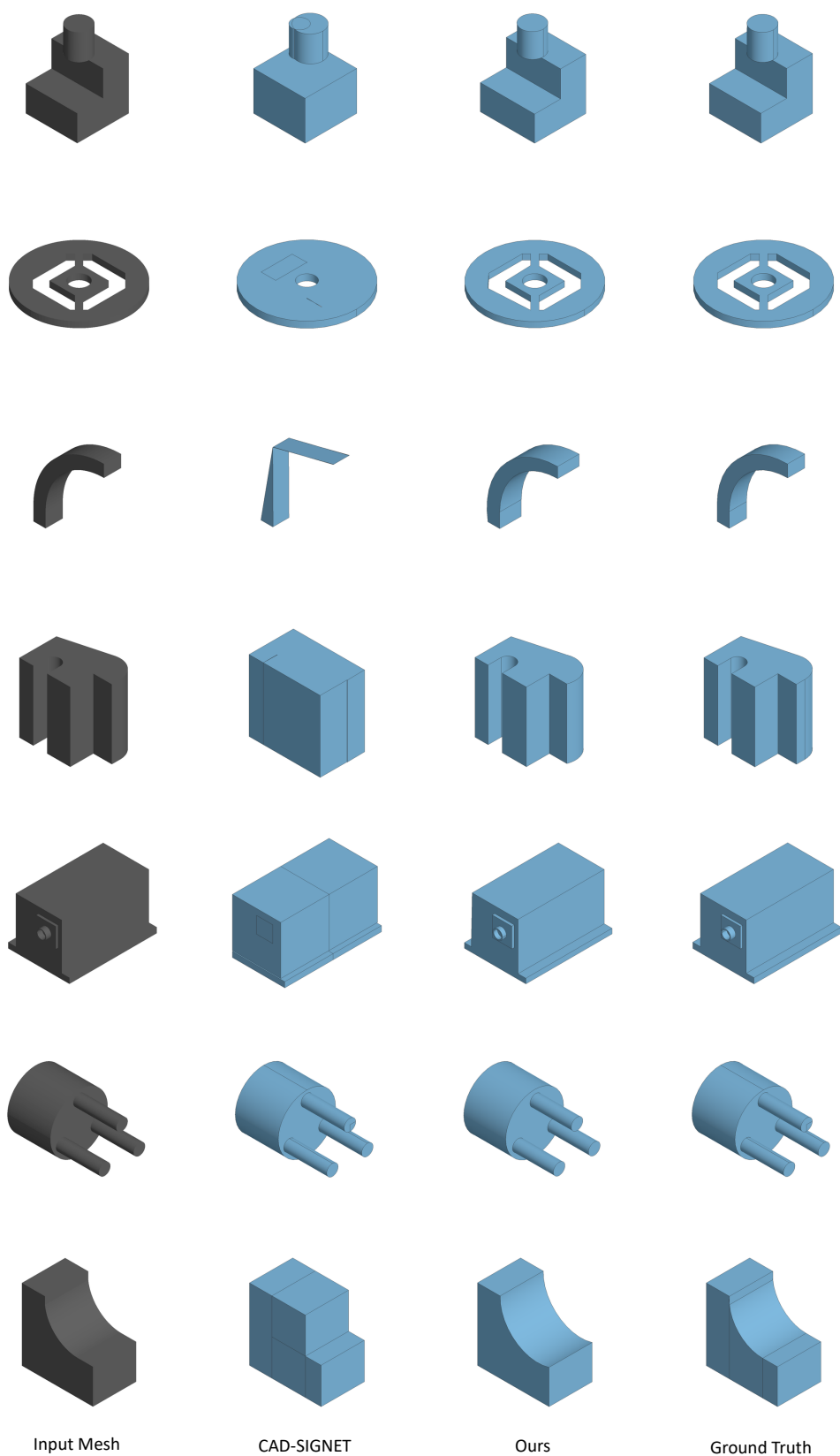


Figure 7: More qualitative comparisons with [4] on DeepCAD dataset.

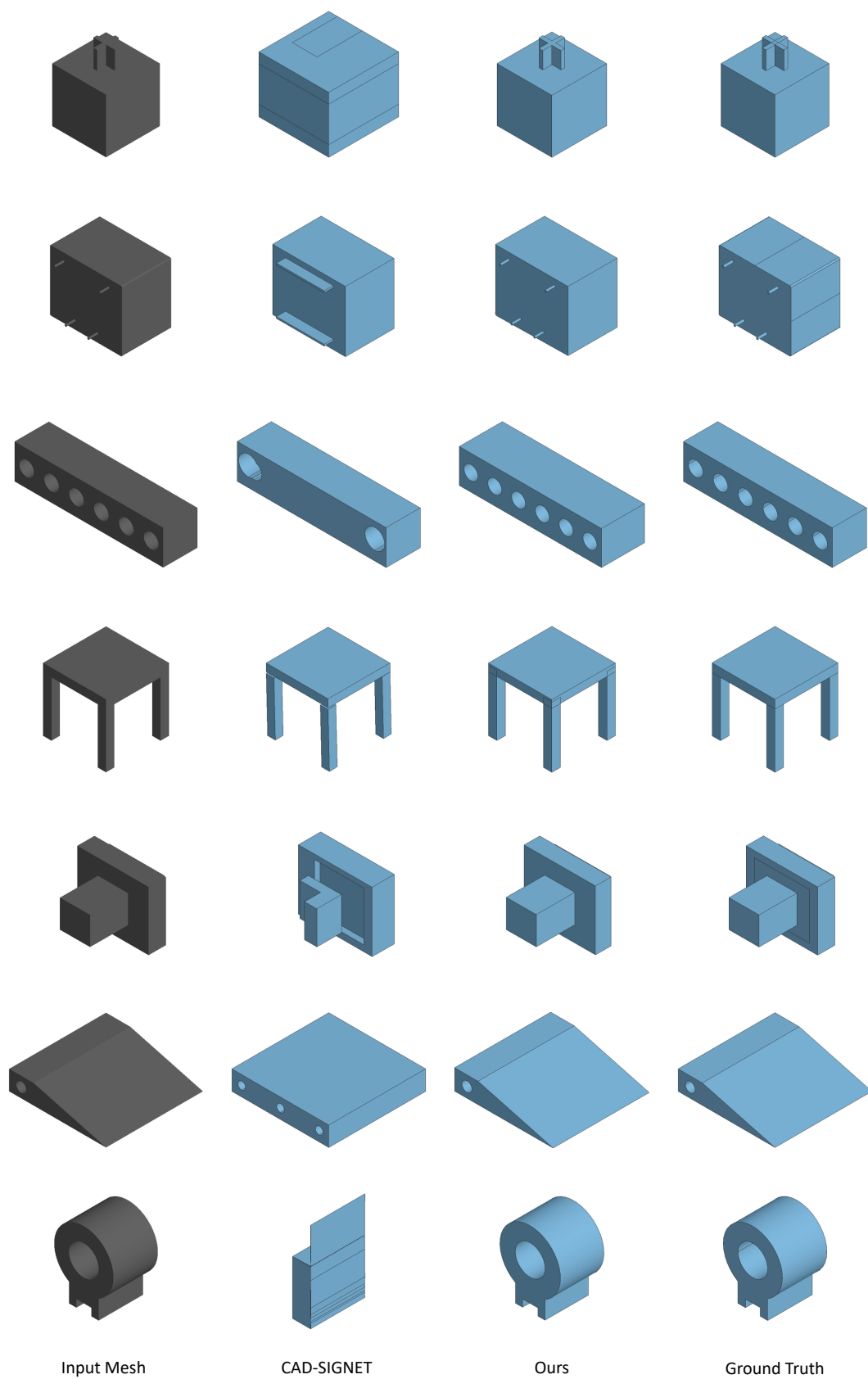


Figure 8: More qualitative comparisons with [4] on DeepCAD dataset.

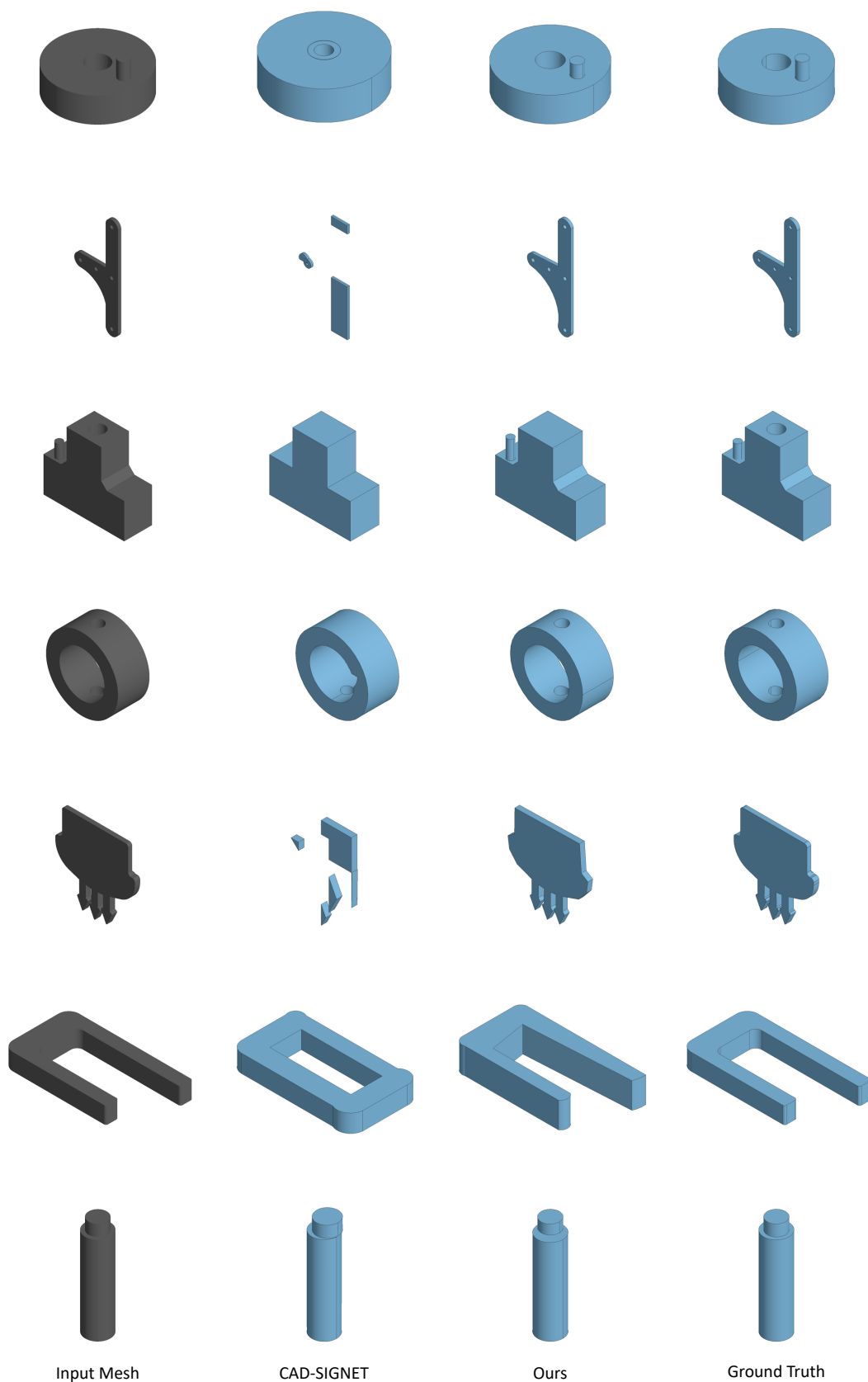


Figure 9: More qualitative comparisons with [4] on Fusion360 dataset.

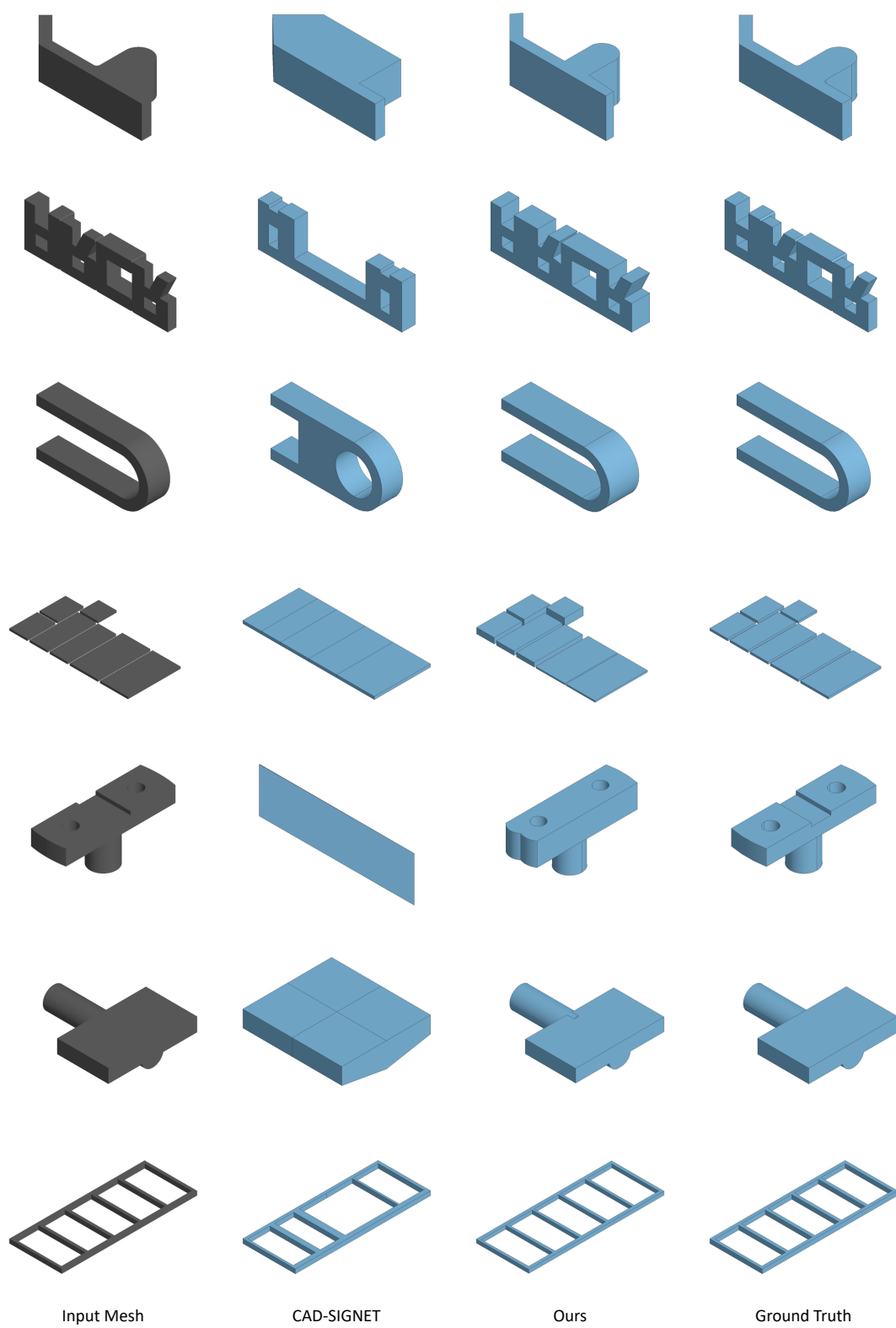


Figure 10: More qualitative comparisons with [4] on Fusion360 dataset.

References

- [1] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *ICCV*, 2021.
- [2] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *ICMLW*, 2020.
- [3] FreeCAD Community. Freecad, 2025. URL <https://www.freecad.org>.
- [4] Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *CVPR*, 2024.
- [5] Weijian Ma, Shuaiqi Chen, Yunzhong Lou, Xueyang Li, and Xiangdong Zhou. Draw step by step: Reconstructing cad construction sequences from point clouds via multimodal diffusion. In *CVPR*, 2024.
- [6] Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *CVPR*, 2022.
- [7] Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *CVPR*, 2023.
- [8] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. In *ECCV*, 2022.
- [9] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. In *ICLR*, 2022.
- [10] Ahmet Serdar Karadeniz, Dimitrios Mallis, Nesryne Mejri, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Davinci: A single-stage architecture for constrained cad sketch inference. In *BMVC*, 2024.
- [11] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.