Point2Cyl: Reverse Engineering 3D Objects
from Point Clouds to Extrusion Cylinders

Mikaela Angelina Uy[*1]   Yen-Yu Chang[*1]   Minhyuk Sung[2]   Purvi Goel[1]

Joseph Lambourne[3]   Tolga Birdal[1]   Leonidas Guibas[1]

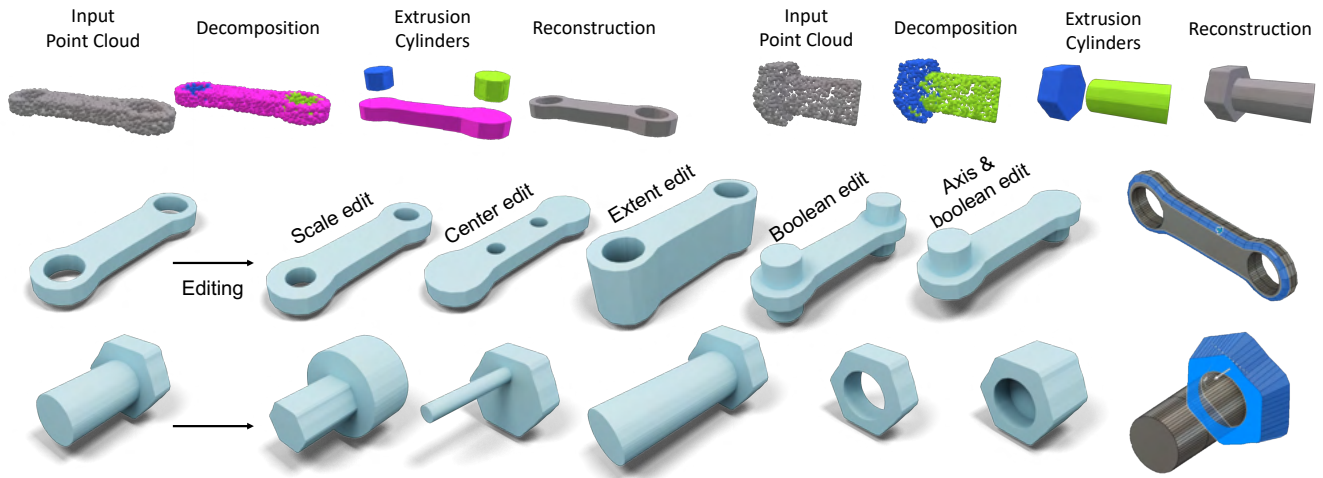[1]Stanford University   [2]KAIST   [3]Autodesk Research

Figure 1. **Overview**. **Point2Cyl** takes a raw point cloud as input and decomposes it into extrusion cylinders while predicting all parameters including the extrusion axis, extent, and the 2D sketch (first row). The output set of extrusion cylinders can be loaded in CAD software and is editable in various ways thus creating a wide array of variations (second and third rows).

Abstract

*We propose **Point2Cyl**, a supervised network transforming a raw 3D **point** cloud **to** a set of extrusion **cylinders**. Reverse engineering from a raw geometry to a CAD model is an essential task to enable manipulation of the 3D data in shape editing software and thus expand their usages in many downstream applications. Particularly, the form of CAD models having a sequence of extrusion cylinders — a 2D sketch plus an extrusion axis and range — and their boolean combinations is not only widely used in the CAD community/software but also has great expressivity of shapes, compared to having limited types of primitives (e.g., planes, spheres, and cylinders). In this work, we introduce a neural network that solves the extrusion cylinder decomposition problem in a geometry-grounded way by first learning underlying geometric proxies. Precisely, our approach first predicts per-point segmentation, base/barrel labels and normals, then estimates for the underlying extrusion parameters in differentiable and closed-form formulations. Our experiments show that our approach demonstrates the best performance on two recent CAD datasets, Fusion Gallery and DeepCAD, and we further showcase our approach on reverse engineering and editing.*

1. Introduction

Our everyday environments are filled with objects fabricated following a carefully engineered computer-aided design. This makes *reverse engineering in the wild* a vital workflow in situations where copies or variations of a physical object are required, but the corresponding CAD model is not available [51]. This situation often occurs when repairing machinery or digitizing objects manufactured in the pre-digital era [11]. To this end, an object is first scanned using a 3D sensor producing a point cloud and later decomposed into a set of consistent primitives or surfaces which could be parsed by existing shape modelers such as Fusion360 [3] or SolidWorks [45]. However, at the user level, a CAD model is designed as a sequence of operations, where the designer first draws a planar 2D *sketch* as a closed curve and later *extrudes* it into a 3D solid [55] (*cf*. Fig. 2). These *Extrusion Cylinders* can then be combined through *boolean* operations.

As this modeling paradigm is hard to summarize using traditional primitives such as planes or cylinders, we set off to ask: *how can we reverse engineer point clouds into primi-*

---

[0]* denotes equal contribution

tives that are interpretable and usable in the modeling process of CAD designers? Traditional approaches answer this question by following a three-step procedure where (i) the point cloud is first converted into a mesh, (ii) subsequently explained by a collection of trimmed parametric surfaces resulting in a watertight solid (a.k.a. boundary representation or B-Rep) [7], and (iii) a CAD program, which could generate the input B-Rep is inferred [56]. Recent trends in fitting primitives to point clouds [9, 24, 46] can bypass the initial meshing stage, but they either assume a finite set of fixed primitives, *e.g.*, planes, cylinders, cones [16, 24, 48] or output a disjoint set of primitives yet to be stitched [8, 33] and thus cannot allow, for example, convenient *shape editing* or *variations*. Note that geometric primitives have unique parameterizations and hence cannot be handled by general 3D model detection pipelines like [27, 35]. As such, the presented problem is much finer-grained than explaining a scene with a retrieved set of CAD models, as done in [2, 50]. Finally, both of the problems we address, *CAD model reconstruction from point sets* and *user editing of CAD shapes* are cast as future work by the recent CAD generative model, DeepCAD [55].

To be able to achieve a geometry-grounded and editable reconstruction, within a CAD-grammar, we propose to cast the problem as a decomposition task into *Extrusion Cylinders*. Our novel approach views *Extrusion Cylinder* as a parametric *primitive* can jointly represent a set of sketch-extrude operations and hence is suitable for representing CAD models. We learn to decompose a given raw point cloud into Extrusion Cylinder instances. In particular, our neural network learns to predict per-point *extrusion instance segmentation*, *surface normal*, and *base/barrel membership*[1]. Given this decomposition, we show how to solve for each primitive's underlying parameters, including the extrusion *axis*, *center*, *sketch*, sketch *scale*, and extrusion *extent*, all in a geometry-grounded way. In return, as shown in Fig. 1, these enable us to reverse engineer the point cloud into an editable 3D CAD model in a format that is directly consumable by existing CAD modelers [1, 3], allowing for further variation creation, *e.g.*, adding fillets/chamfers, modifying the sketch and varying parameters such as center and extents.

In summary, our contributions are the following:

1. We introduce a novel approach that casts the 3D reconstruction task as a Extrusion Cylinder decomposition problem, making it well-suited for CAD modeling.
2. We architect a neural network that decompose an input point cloud into a set of Extrusion Cylinders by learning geometric proxies, which can then used to estimate the extrusion parameters in differentiable, closed-form formulations.
3. We validate our approach quantitatively and qualitatively on two existing CAD datasets, Fusion Gallery [54] and

---
[1]Here, barrel means the cylinder side, the surface swept by the sketch.
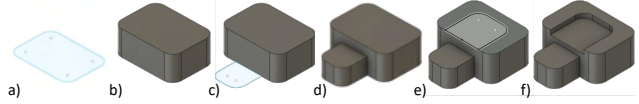

Figure 2. Solid model creation as a sequence of sketch-extrude operations. **a**) Initial sketch. **b**) Volume extruded from that sketch. **c**) A second sketch. **d**) The second sketch is extruded and the boolean union of the two extruded volumes is created. **e**) A third sketch. **f**) The final sketch extruded and subtracted from the solid.

DeepCAD [55], surpassing baselines, and further showcase its applications on reconstruction and shape editing.

Our project page can be found at point2cyl.github.io.

2. Related work

Primitive fitting to point clouds. In vision literature, the primitive fitting and object decomposition have been investigated for decades with diverse types of primitives. The simplest forms of primitives are planes, which have attracted significant attention as they are omnipresent in our environments [10, 15, 17, 28, 47]. More general types of primitives were also explored in the decomposition with RANSAC [25, 39, 49] and region-growing [30] approaches. The local Hough transform of Drost and Ilic [16] showed how the detection of primitives can be made more efficient by considering the local voting spaces. This idea is quickly extended to handle a wider range of primitives [9, 46]. Conic sections are also of special interest as they allow for an infinite set of variations [4, 8, 18, 29].

The proliferation of deep learning has steered researchers to data-driven frameworks [26]. Li *et al*. [24] is the first to use a supervised learning method for the primitive fitting, and Paschalidou *et al*. [34] and Sharma *et al*. [44] extended the idea to fit superquadrics and B-spline surfaces to a point cloud, respectively. However, all these methods assume a fixed set of primitives and none of them handle our Extrusion Cylinders in their fittings.

Reverse engineering in CAD world. In CAD and graphics communities, reverse engineering is a well studied problem with many traditional algorithms explored over the years [12, 51]. Benko *et al*. [7] described a commonly used procedure for segmenting triangle meshes by analysing the pattern which the normal vectors form on the Gaussian sphere. This allows planes, cylinders, cones and doubly curves surfaces to be identified. Following segmentation, these primitives can be fitted using geometric constraints [6].

It is only with the recent availability of large B-Rep, sketch and construction sequence datasets [22, 40, 54, 55] that the machine learning community has become interested in the generation of CAD data [19, 23, 31, 40, 41, 44, 52–56]. Wang *et al*. [52] showed results for identifying keypoints in a point cloud suitable for fitting with parametric curves.

Sharma *et al.* [44] decomposed a point cloud into patches suitable for fitting with parametric surfaces. In both cases the difficult task of combining individual curves or surface to build watertight solid models which can be worked with in CAD modelers is not addressed. Ganin *et al.* [19] use a transformer model to generate 2D sketches. While their model can be conditioned on 2D image data, this data must be a sequence raster images of the individual curves which build up the model – a sequential breakdown not available in the reverse engineering setting. Tian *et al.* [48] showed how shapes can be interpreted as programs represented using an RNN and voxel based "neural renderer".

Most relevant to us is DeepCAD [55], which uncondition-ally generated entire 3D CAD models using a transformer based method. While DeepCAD demonstrates excellent results when auto-encoding in the "program space", it leaves the guidance of the generated shape using point clouds for "future applications", which is what we address in this work.

3. The Extrusion Cylinder

Before delving into the methodological specifics, we first define the Extrusion Cylinder, a *primitive* that gives us the flexibility of creating any shape from arbitrary closed loops, by composing them through a series of boolean operations, mirroring the CAD design process. The extrusion cylinder, represent solid volumes in contrast to existing surface-based works [24, 43], and is parametric as defined below. Our definitions are illustrated in Fig. 3. We further present closed-form and differentiable formulations to recover the extrusion parameters from points.

Definition 1 (Sketch and profile). *We consider a non-self intersecting, finite area, closed loop and normalized 2D* **sketch** $\tilde{\mathbf{S}} = \{p(q(t)) \in \mathbb{R}^2 | t \in [0,1], p(q(0)) = p(q(1))\}$, *for continuous functions* $q : [0,1] \to \mathbb{R}$ *and* $p : \mathbb{R} \to \mathbb{R}^2$. *The area enclosed by* $\tilde{\mathbf{S}}$ *is often called a* **profile**.

Definition 2 (Sketch plane). *We also define the plane containing* $\tilde{\mathbf{S}}$, *parameterized by the center* $\mathbf{c} \in \mathbb{R}^3$, *and a normal along the axis,* $\mathbf{e} \in \mathbb{S}^2$.

Note that the sketch $\tilde{\mathbf{S}}$ defines a profile on the sketch plane parameterized by $(\mathbf{c}, \mathbf{e})$ without ambiguity.

Definition 3 (Extrusion Cylinder). *In manufacturing,* **extrusion** *is the process of pushing the material forward along a fixed cross-sectional profile to a desired height. In our context, we use extrusions to parameterize our primitive, the extrusion cylinder, by an axis* $\mathbf{e} \in \mathbb{S}^2$, *a center* $\mathbf{c} \in \mathbb{R}^3$ *associated to a sketch* $\tilde{\mathbf{S}}$ *scaled by* $s \in \mathbb{R}^2$. *We further introduce the extents* $(r^{min}, r^{max}) \in (\mathbb{R} \times \mathbb{R})$ *defining the extrusion* $E = (\mathbf{e}, \mathbf{c}, \tilde{\mathbf{S}}, s, r^{min}, r^{max})$.
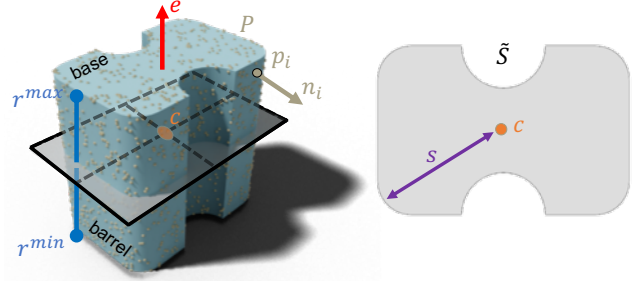

Figure 3. An illustration of our **extrusion cylinder.**

Compared to existing primitive fitting [24,43] or CSG [21, 38, 42] works, we do not assume a finite set of primitives, and instead consider the building blocks to be an arbitrary sketch-extrude operation applied on any closed, non-self-intersecting 2D loop. As such, our extrusion cylinder constitutes a typical building block for CAD pipelines as those used in solid modellers. As we work with point clouds inputs, we now define an attribute associated with points along the *surface* or the *boundary* of the extrusion cylinder.

Definition 4 (Base & Barrel). *We classify points along the surface of an extrusion cylinder as being base or barrel points. Base points are points that lie on the plane at either extents of the extrusion cylinder, while barrel points are points that lie along the "sides" of the extrusion cylinder. Hence, the surface normals of base/barrel points are parallel/perpendicular to the extrusion axis* $\mathbf{e}$.

For any point incident to the boundary of an extrusion cylinder $(\mathbf{p}_i \in \mathbb{R}^3) \in E$, *we represent this attribute by* b_i. $b_i = 0$ *for the barrel and* $b_i = 1$ *for bases. With these, it is easy to verify that for an extrusion axis* $\mathbf{e} \in \mathbb{S}^2$:

$$\begin{cases} b_i = 0 & : & \mathbf{n}_i \perp \mathbf{e} & \mapsto & \mathbf{n}_i^\top \mathbf{e} = 0 \\ b_i = 1 & : & \mathbf{n}_i /\!/ \mathbf{e} & \mapsto & \mathbf{n}_i^\top \mathbf{e} = \pm 1. \end{cases} \quad (1)$$

where $\mathbf{n}_i \in \mathbb{S}^2$ *is the surface normal evaluated at point* $\mathbf{p}_i$.

Recovering extrusion cylinders from points. We now discuss how to recover the parameters of an extrusion cylinder E from a set of points $\mathbf{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^N$ and corresponding normals $\mathbf{N} = \{\mathbf{n}_i \in \mathbb{S}^2\}_{i=1}^N$ incident to E. We let $\mathbf{P}_{\text{base}}, \mathbf{P}_{\text{barr}} \subset \mathbf{P}$ denote base and barrel points of $\mathbf{P}$, respectively, where $\mathbf{P} = \mathbf{P}_{\text{base}} \cup \mathbf{P}_{\text{barr}}$. The **center of the extrusion** ($\hat{\mathbf{c}}$) is the simplest and can be estimated by the taking the mean of all the barrel points of $\mathbf{P}$.

The rest of the parameters depend upon the extrusion axis, for which we give the following algorithm:

Theorem 1 (Recovering **extrusion axis** from points). *For a set of points on an extrusion cylinder, the optimal extrusion axis is given by* [3]:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}, ||\mathbf{e}||=1}{\arg\min} \mathbf{e}^\top \mathbf{H} \mathbf{e}, \quad (2)$$

---

where $\mathbf{H} = (\mathbf{N}_{barr}^{\top}\mathbf{N}_{barr} - \mathbf{N}_{base}^{\top}\mathbf{N}_{base})$, *while* $\mathbf{N}_{base} \in \mathbb{R}^{N_{base}\times 3} \subset \mathbf{N}$ *and* $\mathbf{N}_{barr} \in \mathbb{R}^{N_{barr}\times 3} \subset \mathbf{N}$ *denote the corresponding normals for the base/barrel points belonging to the extrusion cylinder. The solution can be obtained by the eigenvector corresponding to the smallest eigenvalue of* $\mathbf{H}$.

Proof. For an oriented point cloud $(\mathbf{P}, \mathbf{N})$, we could write the relations in Eq (1) as $\mathbf{N}_{barr}\mathbf{e} = \mathbf{0}$ and $\mathbf{N}_{base}\mathbf{e} = \pm\mathbf{1}$, where matrices $\mathbf{N}_{base}$ and $\mathbf{N}_{barr}$ correspond to the point normals belonging to base ($\mathbf{P}_{base} \subset \mathbf{P}$) and barrel ($\mathbf{P}_{barr} \subset \mathbf{P}$) points, respectively. Given that the scalar product for normalized vectors is bounded $[0, 1]$, for $\mathbf{e}$ to be a proper extrusion direction, $\|\mathbf{N}_{base}\mathbf{e}\| \to \max$ and $\|\mathbf{N}_{barr}\mathbf{e}\| \to \min$ must hold, hence:

$$\hat{\mathbf{e}} = \begin{cases} \arg\max_{\mathbf{e},\|\mathbf{e}\|=1} \mathbf{e}^{\top}\mathbf{N}_{base}^{\top}\mathbf{N}_{base}\mathbf{e} & \text{for } \mathbf{N}_{base}\mathbf{e} = \pm\mathbf{1} \\ \arg\min_{\mathbf{e},\|\mathbf{e}\|=1} \mathbf{e}^{\top}\mathbf{N}_{barr}^{\top}\mathbf{N}_{barr}\mathbf{e} & \text{for } \mathbf{N}_{barr}\mathbf{e} = \mathbf{0}. \end{cases}$$

Negating the first and combining both objectives, we get:

$$\hat{\mathbf{e}} = \arg\min_{\mathbf{e},\|\mathbf{e}\|=1} \mathbf{e}^{\top}(\mathbf{N}_{barr}^{\top}\mathbf{N}_{barr} - \mathbf{N}_{base}^{\top}\mathbf{N}_{base})\mathbf{e}. \qquad \square$$

For cases, where the data is contaminated with outlier points non-incident to the extrusion cylinder, we can further introduce a per-point weight factor $\phi \in \mathbb{R}^N$, controlling the contribution of each of the normals in $\mathbf{N}$.

Theorem 2 (Weighted recovery of **extrusion axis** from points)**.** *For a general weighted point set, the optimal **extrusion axis** is given by* $\hat{\mathbf{e}} = \arg\min_{\mathbf{e},\|\mathbf{e}\|=1}(\mathbf{e}^{\top}\mathbf{H}_{\boldsymbol{\Phi}}\mathbf{e})$, *where:*

$$\mathbf{H}_{\boldsymbol{\Phi}} = \mathbf{N}^{\top}\boldsymbol{\Phi}_{barr}^{\top}\boldsymbol{\Phi}_{barr}\mathbf{N} - \mathbf{N}^{\top}\boldsymbol{\Phi}_{base}^{\top}\boldsymbol{\Phi}_{base}\mathbf{N}. \quad (3)$$

where $\boldsymbol{\Phi}_{barr} = \text{diag}(\phi_{barr}), \boldsymbol{\Phi}_{base} = \text{diag}(\phi_{base}) \in \mathbf{R}^{N\times N}$. ϕ_{barr}/ϕ_{base} *indicate the barrel/base weights assigned to all points, respectively. The solution is again given by the eigenvector corresponding to the smallest eigenvalue of* $\mathbf{H}_{\boldsymbol{\Phi}}$.

Proof sketch. $\mathbf{H}_{\boldsymbol{\Phi}}$ is a direct modification of $\mathbf{H}$ incorporating the weights. Full proof is given in our supplementary.

The remainder of the parameters can be estimated as follows. We first define an operator $\Pi(\cdot)$: $\mathbf{q}_i = \Pi(\mathbf{p}_i, \bar{\mathbf{e}}, \bar{\mathbf{c}})$, takes a point ($\mathbf{p}_i \in \mathbb{R}^3$) to ($\mathbf{q}_i \in \mathbb{R}^2$), by aligning $\bar{\mathbf{e}}$ to the z-axis, projecting $\mathbf{p}_i, \mathbf{c}$ onto the $xy-$plane, then centering $\bar{\mathbf{c}}$ at the origin. We define the operator $\boldsymbol{\Pi}(\cdot)$ that does the same mapping for a 3D point cloud $\mathbf{P}$. Hence, an *unnormalized* sketch can be approximated by a 2D point cloud $\boldsymbol{\Pi}(\mathbf{P}_{barr}, \hat{\mathbf{e}}, \hat{\mathbf{c}})$, and from here, the **extrusion scale** $\hat{s}$ can be computed by taking the distance from the farthest point to the origin. This then gives us the *normalized* sketch approximated by $\bar{\mathbf{S}} = \hat{s}_i\boldsymbol{\Pi}(\mathbf{P}_{barr}, \hat{\mathbf{e}}, \hat{\mathbf{c}})$.

The **extrusion extent** is estimated similarly by calculating the minimum and maximum range of $\mathbf{P}_{barr}$ along $\mathbf{e}$ as

illustrated in Fig 3.[4] Altogether:

$$\hat{s} = \max_{\mathbf{p}_i \in \mathbf{P}_{\text{barr}}} \|\Pi(\mathbf{p}_i, \hat{\mathbf{e}}, \hat{\mathbf{c}})\|_2 \qquad (4)$$

$$\hat{r}^{\min} = \min_{\mathbf{p}_i \in \mathbf{P}_{\text{barr}}}(\hat{\mathbf{e}} \cdot (\mathbf{p}_i - \hat{\mathbf{c}})) \quad \hat{r}^{\max} = \max_{\mathbf{p}_i \in \mathbf{P}_{\text{barr}}}(\hat{\mathbf{e}} \cdot (\mathbf{p}_i - \hat{\mathbf{c}}))$$

Since the extrusion axis we estimate is unoriented, we compute the range by taking the maximum of the absolute values from Eq (4), and extruding that computed extent in both directions, *i.e.* $|\hat{r}_k^{\min}| = |\hat{r}_k^{\max}|, \hat{r}_k^{\max} = -\hat{r}_k^{\min}$. Note that, all of these operations lend themselves to differentiation. This will be useful in the following section, where we learn to decompose point clouds into extrusion cylinder primitives.

4. Reverse Engineering Extrusion Cylinders

Problem setting and overview. We assume that we observe an input point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$. Our goal is to decompose its underlying geometry into a set of extrusion cylinders $\{E_1, E_2, ..., E_K\}$. We propose to solve this sketch-extrude decomposition problem in a geometry-grounded way by first learning the underlying geometric properties as a proxies. Specifically, these proxies include (i) an *instance segmentation* $\hat{\mathbf{W}} \in \mathbb{R}^{N\times K}$ as a per-point membership defining the likelihood of assigning each point to a certain segment $k \in \{1 \ldots K\}$, where each segment is an extrusion cylinder (ii) a *base-barrel segmentation* $\hat{\mathbf{B}} \in [0, 1]^{N\times 2}$, instantiating to $\hat{\mathbf{B}}_{:,0} == \mathbf{0}$ for barrel points, $\hat{\mathbf{B}}_{:,1} == \mathbf{1}$ for the base, and (iii) the surface normals ($\hat{\mathbf{N}} \in \mathbb{R}^{N\times 3}$).

In particular, we model our function approximator $\mathcal{G}_{\boldsymbol{\theta}} : \mathbf{P} \mapsto (\hat{\mathbf{M}}, \hat{\mathbf{N}})$ as a neural network, whose architecture will be precised in Sec. 4.3. Each entry in the output $\hat{\mathbf{M}} \in \mathbf{R}^{N\times 2K}$ indicates the likelihood of each point to belong to the base or the barrel of a particular segment.

4.1. Inferring Extrusion Cylinder Parameters

Now given predicted geometric proxies $(\hat{\mathbf{M}}, \hat{\mathbf{N}})$, we establish a differentiable and closed-form formulation to estimate other extrusion parameters(Sec. 3). $\hat{\mathbf{M}}$ compactly and jointly combines the predicted probability of a point 1) being either a base or a barrel, and 2) belonging to a certain segment. We then apply a row-wise softmax turning $\hat{\mathbf{M}}$ into a row-stochastic matrix whose i^{th} row indicates the belonging of point $\mathbf{p_i}$ to one of the $2K$-classes, *i.e.* $\hat{\mathbf{M}}_{i,:} = (\mathbb{P}(\mathbf{p_i} \in \mathbf{P}_{\text{barr}}^0), \mathbb{P}(\mathbf{p_i} \in \mathbf{P}_{\text{base}}^0), \ldots, \mathbb{P}(\mathbf{p_i} \in \mathbf{P}_{\text{barr}}^K), \mathbb{P}(\mathbf{p_i} \in \mathbf{P}_{\text{base}}^K))$ [5]. We can then recover $\hat{\mathbf{W}}$ by summing every two consecutive columns *i.e.* $\hat{\mathbf{W}}_{:,j} = \hat{\mathbf{M}}_{:,2j} + \hat{\mathbf{M}}_{:,2j+1} \forall j$. Similarly, $\hat{\mathbf{B}}$ can be obtained by summing up all odd/even columns of $\hat{\mathbf{M}}$, *i.e.* $\hat{\mathbf{B}}_{:,0} = \sum_i \hat{\mathbf{M}}_{:,2j}$ and $\hat{\mathbf{B}}_{:,1} = \sum_i \hat{\mathbf{M}}_{:,2j+1}$.

---

[4]Only barrel points are used to calculate extent as some extrusion cylinders have no base points.

[5]$\mathbb{P}$ here denotes probability, and superscript denotes the segment index.

Theorem 3. *Matrix $\hat{\mathbf{M}}$ cannot be uniquely recovered from $\hat{\mathbf{W}}$ and $\hat{\mathbf{B}}$.*

We refer the reader to our supplementary material for the full proof. Thm. 3 guarantees that $\mathbf{M}$ is a rather compact parameterization of the number of unknowns *i.e.* it does not suffice to learn $\mathbf{W}$ and $\mathbf{B}$, individually. Once predicted, we can directly use $\hat{\mathbf{M}}$ in order to solve for the parameters of each extrusion cylinder. To this end, we set:

$$\boldsymbol{\Phi}^k_{\text{barr}} = \text{diag}(\hat{\mathbf{M}}_{:,2k}) \quad \boldsymbol{\Phi}^k_{\text{base}} = \text{diag}(\hat{\mathbf{M}}_{:,2k+1}), \quad (5)$$

$\forall k \in [1.K]$, and use them (along with the surface normals $\hat{\mathbf{N}}$ as the weights for the algorithm presented in Thm. 2 to obtain $\hat{\mathbf{e}}_k$ for each segment k[6]. Then, the rest of the parameters are estimated individually for each segment as explained in Sec. 3, *e.g.* Eq (4).

Inferring sketches. Finally, we show how to predict the sketch representation $\tilde{\mathbf{S}}$. We first project the barrel points of each segment k onto the plane defined by $(\hat{\mathbf{c}}_k, \hat{\mathbf{e}}_k)$ and scale by $\hat{s}_k$. This results in a 2D point cloud $\hat{\mathbf{S}}_k \in \mathbb{R}^{N^k_{\text{barr}} \times 2} = s_k\Pi(\hat{\mathbf{P}}^k_{\text{barr}}, \hat{\mathbf{e}}_k, \hat{\mathbf{c}}_k)$[7]. This creates a representational discrepancy when compared to Dfn. 1 where the loops must be closed. A naïve approach could attempt to summarize the points into 2D primitives *e.g.* line segments and arcs. However, (i) guaranteeing that the output sketch is closed and non-self intersecting is hard, and (ii) there can be multiple approximations of primitives for the same 2D sketch, hampering the learnability. Instead, inspired by the recent deep implicit works [5, 32], we represent the sketch implicitly, by learning the parameters $\boldsymbol{\beta}$ of an *encoder function* $f_{\boldsymbol{\beta}}(\hat{\mathbf{S}}_k) \in \mathbb{R}^D$ that maps the 2D point cloud into a global, normalized *sketch latent space*. This latent code acts as the condition of a *decoder* $\mathcal{S} : (\mathbb{R}^D \times \mathbb{R}^2) \to \mathbb{R}$ mapping $(\mathbf{r} \in \mathbb{R}^2)$ to its signed distance value to the underlying normalized sketch $\tilde{\mathbf{S}}_k$: $d(\tilde{\mathbf{S}}_k, \mathbf{r}) \approx \mathcal{S}(f(\hat{\mathbf{S}}_k), \mathbf{r})$. Here, d is the distance between $\mathbf{r}$ and ground truth *sketch curve* of the segment $\tilde{\mathbf{S}}_k$, approximated by implicit function $\mathcal{S}$ that is conditioned on the encoded 2D *point cloud sketch approximation* $f(\hat{\mathbf{S}}_k)$. During inference, marching squares can be ran on the resulting field to obtain the closed sketch curves. The training of f, $\mathcal{S}$ and $\mathcal{G}_{\boldsymbol{\theta}}$ will be specified below.

4.2. Training

Assuming the availability of ground truth (GT) labels, we train the parameters $\boldsymbol{\theta}$ of $\mathcal{G}_{\boldsymbol{\theta}}$ using a multi-task, non-convex objective composed of segmentation (seg), base-barrel classification (bb), normal (norm) and sketch regularization losses:

$$\mathcal{L} = \mathcal{L}_{\text{seg}} + \lambda_{\text{bb}}\mathcal{L}_{\text{bb}} + \lambda_{\text{norm}}\mathcal{L}_{\text{norm}} + \lambda_{\text{sketch}}\mathcal{L}_{\text{sketch}}, \quad (6)$$

---

[6]The weights used here are the probabilities that each point is a base/barrel point of extrusion segment k

[7]Similarly, we project the 3D normals of each point onto the same sketch plane endowing the 2D sketch with 2D normals.

where $\lambda_{\text{bb}} = \lambda_{\text{norm}} = \lambda_{\text{sketch}} = 1$. In what follows, we define and detail each of the loss terms.

Normal estimation loss ($\mathcal{L}_{\text{norm}}$). We begin by estimating per-point unoriented normals $\hat{\mathbf{N}}^{N \times 3}$ for the given input point cloud $\mathbf{P}$. and penalize for absolute cosine distance between the predicted and GT normals:

$$\mathcal{L}_{\text{norm}} = \sum_{i=1}^{N}(1 - |\hat{\mathbf{N}}^{\top}_{i,:}\mathbf{N}_{i,:}|). \quad (7)$$

Extrusion cylinder segmentation loss ($\mathcal{L}_{\text{seg}}$). Points belonging to the same extrusion cylinder segment are assumed to be created by the same sketch-extrusion block. Note that, our problem does not admit a unique solution to the ordering of the segments as there could be multiple possible orderings of the sketch-extrude blocks that can yield the same output geometry. We instead predict a set of unordered segments and use Hungarian matching to find the best one-to-one matching with the ground truth extrusion cylinder segments, by computing Relaxed Intersection over Union (RIoU) between the predicted and ground truth segmentation:

$$\text{RIoU}(\mathbf{U}, \mathbf{V}) = \frac{\mathbf{U}^{\top}\mathbf{V}}{||\mathbf{U}||_1 + ||\mathbf{V}||_1 - \mathbf{U}^{\top}\mathbf{V}}. \quad (8)$$

Our final loss then maximizes the mean RIoU between the predicted and ground truth extrusion cylinder segments:

$$\mathcal{L}_{\text{seg}} = \frac{1}{N}\sum_{i=1}^{N}(1 - \text{RIoU}(\hat{\mathbf{W}}_{i,:}, \mathbf{W}_{i,:})). \quad (9)$$

This also maintains the pipeline to be differentiable almost everywhere, and hence can be trained end-to-end. From here on, our notations assume a reordering such that the predicted and ground truth extrusion cylinders are in correspondence.

Base-and-Barrel classification loss ($\mathcal{L}_{\text{bb}}$). Retrieving $\hat{\mathbf{B}}$ from $\hat{\mathbf{M}}$ allows us to further impose a binary cross-entropy loss between all the base-barrel memberships and their corresponding GT labels:

$$\mathcal{L}_{\text{bb}} = \sum_{i=1}^{N}\sum_{j=1}^{2}\hat{\mathbf{B}}_{i,j}\log(\mathbf{B}_{i,j}). \quad (10)$$

Sketch consistency loss ($\mathcal{L}_{\text{sketch}}$). To ensure that the predicted parameters can yield meaningful sketches, we introduce the regularizer $\mathcal{L}_{\text{sketch}}$. We first describe the the training process for $f_{\boldsymbol{\beta}}$ and $\mathcal{S}$ before defining $\mathcal{L}_{\text{sketch}}$.

Since it is difficult to obtain direct supervision for GT signed distance fields, we propose to learn the implicit curve representation directly from *raw* 2D point clouds. We use ground truth barrel points for each segment ($\mathbf{P}^k_{\text{barr}}$) of the models and its extrusion parameters to obtain clean sketches $\mathbf{S}_k \in \mathbb{R}^{N_k \times 2} = s_k\mathbf{\Pi}(\mathbf{P}^k_{\text{barr}}, \mathbf{e}_k, \mathbf{c}_k))$ to train $f_{\boldsymbol{\beta}}$ and $\mathcal{S}$. Yet, our sketches can be created by positive or negative operations

and we cannot uniquely determine the sign of the surface normals. Thus, we use an *unsigned* variant of IGR [20], to learn $f_{\boldsymbol{\beta}}$ and $\mathcal{S}$. The loss for a sketch point cloud $\{\mathbf{s}_i \in \mathbb{R}^2 \subset \mathbf{S}_k\}$ [8] with corresponding normals $\{\mathbf{g}_i \in \mathbb{S}^1\}$ is defined as

$$\mathcal{L}_{\mathcal{S}} \triangleq \mathcal{L}_{\text{manifold}} + \lambda_1 \mathcal{L}_{\text{non-manifold}}, \qquad (11)$$

where

$$\mathcal{L}_{\text{manifold}} = \frac{1}{|\mathbf{S}_k|} \sum_{\mathbf{s}_i \in \mathbf{S}_k} (|\mathcal{S}(f_{\boldsymbol{\beta}}(\mathbf{S}_k), \mathbf{s}_i)| \\ + \lambda_2 \min(||\nabla \mathcal{S}(f_{\boldsymbol{\beta}}(\mathbf{S}_k), \mathbf{s}_i) \pm n_i||)). \qquad (12)$$

Note, the second term is different from the original IGR [20] as it considers *unoriented* normals. Moreover,

$$\mathcal{L}_{\text{non-manifold}} = \mathbb{E}_{\mathbf{r}}(||\nabla_{\mathbf{r}} \mathcal{S}(f_{\boldsymbol{\beta}}(\mathbf{S}_k), \mathbf{r})|| - 1), \qquad (13)$$

denotes the Eikonal term and encourages the gradients to have a unit norm everywhere ($\forall \mathbf{r} \in \mathbb{R}^2$). Below assumes that $f_{\boldsymbol{\beta}}$ and $\mathcal{S}$ are trained to convergence, which then allows us to define our sketch regularization loss.

We make the observation that when the decomposition and hence extrusion parameters we predict are close to the GT, the latent embedding of the projected 2D point cloud $\hat{\mathbf{S}}_k = \hat{s}_k \mathbf{\Pi}(\hat{\mathbf{P}}_{\text{barr}}^k, \hat{\mathbf{e}}_k, \hat{\mathbf{c}}_k)$ should be close to the GT sketch embedding $f_{\boldsymbol{\beta}}(\mathbf{S}_k)$, $\hat{\mathbf{P}}_{\text{barr}}^k$ is obtained using $\hat{M}$. Under the light of this *prior*, we train a joint embedding space g together with $\mathcal{G}_{\theta}$ to match the sketch embedding $f_{\boldsymbol{\beta}}$.

$$\mathcal{L}_{\text{sketch}} = ||g(\hat{\mathbf{S}}_k) - f_{\boldsymbol{\beta}}(\mathbf{S}_k)||_2. \qquad (14)$$

As $\mathcal{G}_{\theta}$ and g are jointly optimized, this loss encourages $\mathcal{G}_{\theta}$ to predict extrusion cylinder decompositions that result in parameters that produce sketches close to GT.

4.3. Network Details

We use a Pointnet++ [37] backbone to learn a global 3D point cloud feature. This feature vector is then passed through two separate fully connected branches to obtain instance and base/barrel segmentations $\hat{\mathbf{M}}$ as well as normals $\hat{\mathbf{N}}$. Our network, $\mathcal{G}_{\boldsymbol{\theta}}$ is first trained with segmentation ($\mathcal{L}_{\text{seg}}$), base-barrel classification ($\mathcal{L}_{\text{bb}}$) and normal losses ($\mathcal{L}_{\text{norm}}$). We also separately pre-train the sketch implicit network $\mathcal{S}$ using as input, the ground truth point clouds projected and scaled using the ground truth axis/scale for each separate extrusion segment of all the models. We use the same architecture as IGR [20] for $\mathcal{S}$, and a PointNet [36] encoder for $f_{\boldsymbol{\beta}}$ with latent dimension $D = 256$. We then append both pre-trained networks together train our full model (**Point2Cyl**) using the full loss function $\mathcal{L}$ with $\mathcal{L}_{\text{sketch}}$ defined in Eq (6). g was trained from scratch with the same architecture as $f_{\boldsymbol{\beta}}$. $f_{\boldsymbol{\beta}}$ and $\mathcal{S}$ are kept fixed during the final training. More details can be found in the supplementary.

---

[8]Note that here we define the loss per sketch point cloud of an extrusion segment index k denoted by $\mathbf{S}_k$, for the *ground truth* projected point cloud and $\hat{\mathbf{S}}_k$ for the predicted projected point cloud.

5. Results

Datasets. We adapt the two recent CAD datasets, Fusion Gallery [54] and DeepCAD [55], to show the feasibility of our approach. We first preprocess the data to extract a set of extrusion cylinders with their parameters for each solid model. For each model, we randomly sample 8192 points from each underlying normalized and centered mesh to obtain input 3D point clouds. Ground truth sketches are represented as normalized 2D point clouds from randomly sampling 8192 points from the barrels of each extrusion cylinder. We train/test our networks on 4316/1242 models on Fusion and 34910/3087 models on DeepCAD. Details on the preprocessing, extrusion cylinder and sketch extraction are found in our supplementary material.

Evaluation metrics. To the best of our knowledge, we are the firsts to address the problem at hand. Hence, we assess different aspects of our algorithm by introduction a diverse set of evaluation metrics, defined per model, as follows:

- **Segmentation IoU (Seg.)** Assuming that the Hungarian matches reorders the extrusion segment into correspondence with the GT, we define the total RIoU loss: $\frac{1}{K} \sum_{k=1}^K \text{RIoU}(\mathbb{1}(\hat{\mathbf{W}}_{:,k}), \mathbf{W}_{:,k})$, where $\mathbb{1}(\cdot)$ denotes a one-hot conversion.
- **Normal angle error (Norm.)** We consider the agreement of GT and predicted normals: $\frac{1}{N} \sum_{i=1}^N \cos^{-1}(|\hat{\mathbf{N}}_{i,:}^T \mathbf{N}_{i,:}|)$
- **Base/barrel classification accuracy (B.B.)** The amount of correctly predicted base/barrel labels reads: $\frac{1}{N} \sum_{i=1}^N (\mathbb{1}(\hat{\mathbf{B}}_{i,:}) == \mathbf{B}_{i,:})$
- **Extrusion-axis angle error (E.A.)** defines the angular error between GT and predicted axes: $\frac{1}{K} \sum_{k=1}^K \cos^{-1}(|\hat{\mathbf{e}}_{k,:}^T \mathbf{e}_{k,:}|)$
- **Extrusion center error (E.C.)** measures the distance to the GT center: $\frac{1}{K} \sum_{k=1}^K ||\hat{\mathbf{c}}_{k,:} - \mathbf{c}_{k,:}||_2$
- **Per-extrusion cylinder fitting loss (Fit Cyl.)** measures how well the predicted unbounded extrusion cylinder parameters fit the GT extrusion cylinder segments:

$$\frac{1}{K} \sum_{k=1}^K \left(\mathcal{F} \triangleq \sum_{\mathbf{s_i} \in \bar{\mathbf{S}}_k} |\mathcal{S}(f_{\beta}(\hat{\mathbf{S}}_k), \mathbf{s_i})| \right), \qquad (15)$$

where $\bar{\mathbf{S}}_k = \hat{s}_k \mathbf{\Pi}(\mathbf{P}^{\text{barrel}_k}, \hat{\mathbf{e}}_k, \hat{\mathbf{c}}_k)$, which projects per-segment GT barrel points using predicted extrusion parameters. Let us call the inner summation $\mathcal{F}(\mathbf{P}, k)$, which represents the goodness of fit for the kth extrusion.

- **Global fitting loss (Fit Glob.)** measures how at least one (any) of the predicted unbounded extrusion cylinders explain the input per each barrel point:

$$\frac{1}{|\mathbf{P}_{\text{barr}}|} \sum_{\mathbf{p}_i \in \mathbf{P}_{\text{barr}}} \min_{k=\{1,...,K\}} |\mathcal{S}(f_{\beta}(\hat{\mathbf{S}}_k), \mathbf{q}_i)|, \qquad (16)$$

where $\mathbf{q}_i = \hat{s}_k \mathbf{\Pi}(\mathbf{p}_i, \hat{\mathbf{e}}_k, \hat{\mathbf{c}}_k)$.

Final scores are the average of the metrics across all shapes.

Table 1. Quantitative results on Fusion Gallery and DeepCAD datasets.

|		Seg.↑	Norm.(°)↓	B.B.↑	E.A.(°)↓	E.C.↓	Fit Cyl↓	Fit Glob↓
Fusion Gallery	H.V. + $\mathbf{N_J}$	0.409	12.264	0.595	58.868	0.1248	0.1492	0.0683
	D.P.	-	-	-	30.147	0.1426	1.4132	0.4257
	w/o $\mathcal{L}_{\text{sketch}}$ + $\mathbf{N_J}$	0.699	12.264	**0.913**	14.169	0.0729	0.0828	0.0330
	w/o $\mathcal{L}_{\text{sketch}}$	0.699	8.747	**0.913**	9.795	0.0727	0.0826	0.0352
	Ours (**Point2Cyl**)	**0.736**	**8.547**	0.911	**8.137**	**0.0525**	**0.0704**	**0.0305**
DeepCAD	H.V. + $\mathbf{N_J}$	0.540	13.573	0.577	59.785	0.0435	0.1664	0.0459
	D.P.	-	-	-	48.818	0.0716	0.4947	0.4840
	w/o $\mathcal{L}_{\text{sketch}}$ + $\mathbf{N_J}$	0.829	13.573	0.916	10.109	0.0275	0.0856	0.0312
	w/o $\mathcal{L}_{\text{sketch}}$	0.829	8.850	0.916	8.085	0.0273	0.0783	0.0324
	Ours (**Point2Cyl**)	**0.833**	**8.563**	**0.919**	**7.923**	**0.0267**	**0.0758**	**0.0308**
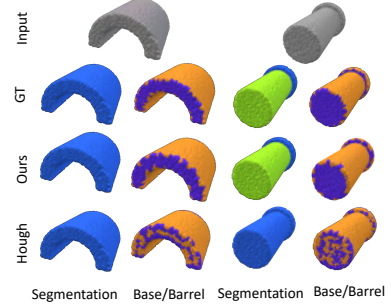
Figure 4. Comparisons with Hough voting baseline.

Implementation and runtime. We implement our network using Pytorch and trained our models for ~ 300 epochs/until convergence. Moreover, our approach has 3.6M parameters, and on a single Titan RTX GPU, it takes 0.41s for each training batch and 0.25s for a single model at inference time.

5.1. Baselines

Classical approaches. While the complexity of the extrusion axis decomposition problem makes it tedious to design handcrafted algorithms, we are still interested in seeing what our data-driven approach could offer over the classical methods. We first compare the point normals learned by Point2Cyl with a strong geometric baseline *jet fitting* [13], ($\mathbf{N_J}$). We then use these jet-normals to vote for the extrusion axes via Hough voting (**H.V.**). To distinguish multiple primitives, we back the voting by a mean-shift mode estimation [14]. Eventually, this baseline produces both the predicted segments as well as their corresponding extrusion axes. Hence, we can compute the per-point base/barrel labels by taking the dot product between point normals and extrusion axes. The other extrusion cylinder parameters can then be computed as described in our method.

Direct prediction (D.P.). We introduce a second baseline, in which a deep neural network that is trained to *directly predict* the set of extrusion cylinder parameters *without* first learning our geometric proxies which are the per point segmentation and normals. As such, it lacks the appropriate geometric inductive biases present in our method. **D.P.** yields K sets of extrusion parameters $\{(\hat{\mathbf{e}}_k), (\hat{\mathbf{c}}_k), \hat{s}_k), g_{\text{DP}}(\hat{\mathbf{S}}_k)\}$[9], with a supervision signal per each parameter. We use Hungarian matching on $\mathcal{F}(\mathbf{P}, k)$, as previously used in Fit Cyl., to sort the predictions in the same order as the ground truth.

The details of both (i) deriving the Hough transform for our primitives and (ii) the architecture of the **D.P.** baseline can be found in the supplementary document.

5.2. Experimental Evaluation

Quantitative evaluations. Tab. 1 shows our quantitative results on both the smaller Fusion Gallery and the big Deep-CAD datasets. Our experiments demonstrate that our approach outperforms the other baselines (top two rows of each sub-block) across all metrics, often by significant margins, *i.e.* $> 15°$ in E.A. Note that as the D.P. method does not produce intermediary proxies, we cannot evaluate those.

In general, we see that even the direct prediction can work better than a simple handcrafted baseline (**H.V.**). This is further illustrated in Fig. 4 (left two columns) which shows a typical case where **H.V.** is able to vote for a reasonable extrusion axis, achieving reasonable base/barrel classification. However, as Fig. 4 (right two columns) illustrates, Hough cannot disambiguate segments that share the same extrusion axis. Such dependencies are hard to model. However, our learning-based approach can handle such cases thanks to its adaptivity to data. This further verifies the difficulty of manually modeling our problem.

Additionally, our results affirm the validity of our geometric biases by revealing that predicting extrusion cylinder primitives through our proxy losses yields more faithful primitives compared to directly regressing the parameters as done by **D.P.**.

Reverse engineering. We propose two frameworks for reconstruction from extrusion cylinders. First, is a volume-based reconstruction engine, which builds a 3D signed distance volume by composing each of the extruded 2D sketch implicit fields from our prediction. The 3D transformation and scale are obtained from the corresponding extrusion cylinder parameters. We also refine our segmentation prediction by a simple post-filtering and use robust methods in estimation of the scale and extent. More on the post-processing details lie in the supplementary. Fig. 5 shows representative examples reconstructed by our predicted extrusion cylinders using our volumetric reconstruction pipeline. The figure also presents the individual extrusion primitives, individually reconstructed. We can see that the predicted extrusion cylinders result from a variety of closed loops, showing that it is different from the standard primitive fitting works where the essential form of the primitive remains intact.

Our second reconstruction method exploits existing CAD modeller software such as Fusion360. To prepare the right input, we first extract the 2D sketch profiles by running

---

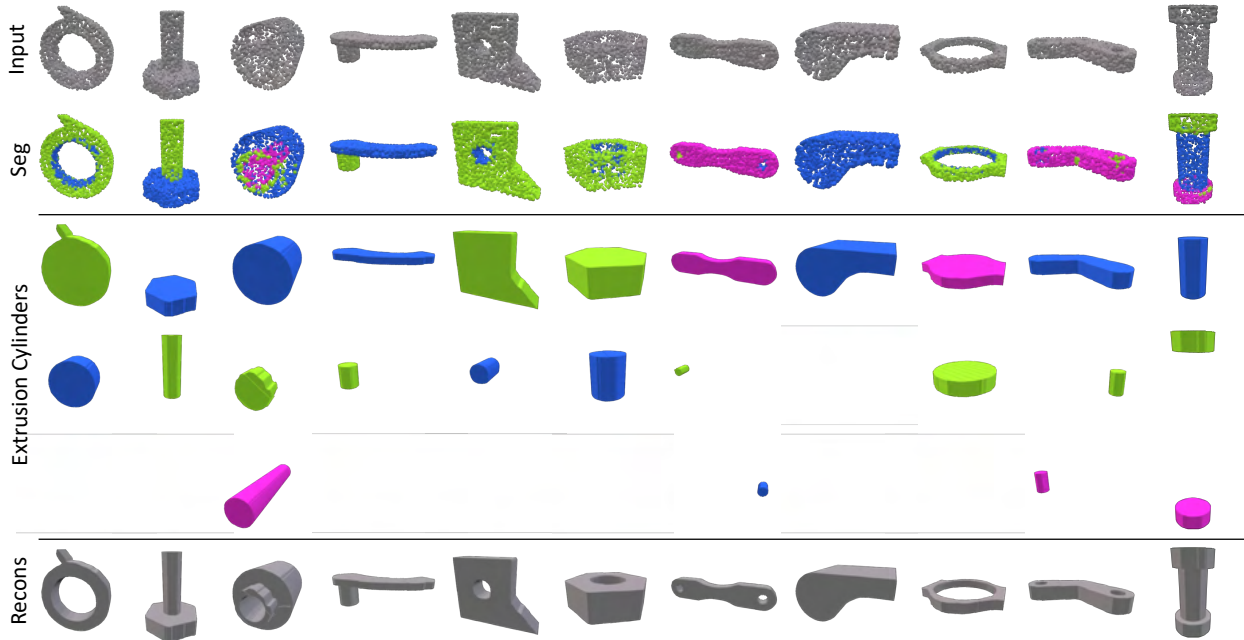[9]g_{DP} is analogous to g in $\mathcal{L}_{\text{sketch}}$

Figure 5. Qualitative examples for reconstruction. Figure shows (top-to-bottom) (1) input point clouds, (2) our predicted segmentation, (3-5) corresponding set of extrusion cylinders and (6) our final reconstruction. This figure also illustrates that individual extrusion cylinders from our decomposition result from a variety of closed loops.

marching squares on the predicted sketch implicits and then define the transformation matrices for each primitive using the extrusion cylinder parameters. Bottom-left most two shapes in Fig. 1 depict such reconstructions.

CAD editing. Once reconstructed using the latter technique, the outputs can be loaded into the existing CAD modellers for further editing such as adding fillets and chamfers. A sequence of such successful edits are shown in Fig. 1 (bottom rows). As shown, we can create different interpretable shape variations from our extrusion cylinders by varying their parameters and/or their operation. Note that, exploring the space of such variations in a controllable manner would be challenging for any existing generative model such as [55].

Ablation studies. Finally, we ablate our approach by leaving out optional components such as the light sketch prior g (w/o $\mathcal{L}_{\text{sketch}}$) or data-driven normal estimation in addition to the sketch prior (w/o $\mathcal{L}_{\text{sketch}} + \mathbf{N_J}$) as shown in Tab. 1. These variants are trained similar to our original network to predict geometric proxies, but without applying the corresponding losses to backpropagate the gradients. Note that, without the sketch prior, the network can not regularize to produce segments that project close to a closed loop sketch, while not learning the normals cannot adaptively correct the issues or biases inherent in the hand-crafted estimation. For these reasons, both of these variants under-perform Point2Cyl. Though, they could still perform reasonable well when compared to the direct prediction or Hough baselines.

6. Conclusion

We present Point2Cyl, for reverse engineering 3D CAD models into primitives interpretable and usable by CAD designers. To solve this challenging discrete-continuous decomposition problem, we first introduced the Extrusion Cylinder and developed its foundations for fitting to point sets. We then proposed differentiable algorithms suitable for a neural architecture, which partitions a point cloud into a set of Extrusion Cylinders. Our network benefited from a set of proxy predictions, which are shown to inject the correct geometric inductive biases. As opposed to standard primitive fitting, the output of Point2Cyl allows for shape variations, and can be directly imported into existing CAD modellers for further reconstruction, visualization and re-usability.

Limitations. Our approach does not make use of a known ordering for the decomposition or reconstruction. This circumvents assembly ambiguities and is beneficial for our training pipeline. Furthermore, each extrusion cylinder primitive can either be a positive or negative volume with respect to the final shape. We also cannot disambiguate certain primitives *e.g.* a box-like shape.

References

[1] Open cascade technology. 2

[2] Raster to vector software: Convert images to cad. 2

[3] Fusion 360: 3d cad, cam, cae & pcb cloud-based software, Aug 2021. 1, 2

[4] James Andrews and Carlo H Séquin. Type-constrained direct fitting of quadric surfaces. *Computer-Aided Design and Applications*, 2014. 2

[5] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 5

[6] Pál Benkö, Géza Kós, Tamás Várady, László Andor, and Ralph Robert Martin. Constrained fitting in reverse engineering. *Comput. Aided Geom. Des.*, 19:173–205, 2002. 2

[7] Pal Benko and Tamas Varady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design*, 36(6):511–523, 2004. 2

[8] Tolga Birdal, Benjamin Busam, Nassir Navab, Slobodan Ilic, and Peter Sturm. A minimalist approach to type-agnostic detection of quadrics in point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3530–3540, 2018. 2

[9] Tolga Birdal, Benjamin Busam, Nassir Navab, Slobodan Ilic, and Peter Sturm. Generic primitive detection in point clouds using novel minimal quadric fits. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1333–1347, 2019. 2

[10] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2, 2011. 2

[11] Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, and Yary Volpe. Reverse engineering modeling methods and tools: a survey. *Computer-Aided Design and Applications*, 15(3):443–464, 2018. 1

[12] Roseline Bénière, Gérard Subsol, Gilles Gesquière, François Le Breton, and William Puech. A comprehensive process of reverse engineering from 3d meshes to cad models. *Computer-Aided Design*, 45(11):1382–1393, 2013. 2

[13] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. 7

[14] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995. 7

[15] T Czerniawski, B Sankaran, M Nahangi, C Haas, and F Leite. 6d dbscan-based segmentation of building point clouds for planar object classification. *Automation in Construction*, 88:44–58, 2018. 2

[16] Bertram Drost and Slobodan Ilic. Local hough transform for 3d primitive detection. In *2015 International Conference on 3D Vision*, pages 398–406. IEEE, 2015. 2

[17] Hao Fang, Florent Lafarge, and Mathieu Desbrun. Planar shape detection at structural scales. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[18] J-M Frahm and Marc Pollefeys. Ransac for (quasi-) degenerate data (qdegsac). In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 453–460. IEEE, 2006. 2

[19] Yaroslav Ganin, Sergey Bartunov, Yujia Li, E. Keller, and Stefano Saliceti. Computer-aided design as language. *ArXiv*, abs/2105.02769, 2021. 2, 3

[20] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *ICML*. 2020. 6

[21] Kacper Kania, Maciej Zięba, and Tomasz Kajdanowicz. Ucsg-net – unsupervised discovering of constructive solid geometry tree. In *NeurIPS*, 2020. 3

[22] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. 2

[23] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Trans. Graph. (Proceedings of SIGGRAPH Asia 2020)*, 39(6):164:1–164:14, 2020. 2

[24] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *CVPR*, 2019. 2, 3

[25] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. In *ACM Transactions on Graphics (TOG)*, 2011. 2

[26] Eric-Tuan Lê, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J. Mitra. Cpfn: Cascaded primitive fitting networks for high resolution point clouds. *ICCV*, 2021. 2

[27] Ishan Misra, Rohit Girdhar, and Armand Joulin. An End-to-End Transformer Model for 3D Object Detection. In *ICCV*, 2021. 2

[28] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, 34(4):103–1, 2015. 2

[29] Thomas Mörwald, Andreas Richtsfeld, Johann Prankl, Michael Zillich, and Markus Vincze. Geometric data abstraction using b-splines for range image segmentation. In *Robotics and Automation, International Conference on*. IEEE, 2013. 2

[30] Sven Oesau, Florent Lafarge, and Pierre Alliez. Planar shape detection and regularization in tandem. In *Computer Graphics Forum*, volume 35, pages 203–215. Wiley Online Library, 2016. 2

[31] Wamiq Reyaz Para, Shariq Farooq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches, 2021. 2

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, June. 5

[33] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10344–10353, 2019. 2

[34] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10344–10353, 2019. 2

[35] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. 2

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 6

[37] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 6

[38] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, et al. Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. In *ICCV*, 2021. 3

[39] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, 2007. 2

[40] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *ICML 2020 Workshop on Object-Oriented Learning*, 2020. 2

[41] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P. Adams. Vitruvion: A generative model of parametric cad sketches, 2021. 2

[42] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *CVPR*, June 2018. 3

[43] Gopal Sharma, Difan Liu, Evangelos Kalogerakis, Subhransu Maji, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In *ECCV*, 2020. 3

[44] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds, 2020. 2, 3

[45] Dassault Systèmes SolidWorks. Solidworks®. *Version Solidworks*, 2005. 1

[46] Christiane Sommer, Yumin Sun, Erik Bylow, and Daniel Cremers. Primitect: Fast continuous hough voting for primitive detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8404–8410. IEEE, 2020. 2

[47] Christiane Sommer, Yumin Sun, Leonidas Guibas, Daniel Cremers, and Tolga Birdal. From planes to

corners: Multi-purpose primitive detection in unorganized 3d point clouds. *IEEE Robotics and Automation Letters*, 5(2):1764–1771, 2020. 2

[48] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Learning to infer and execute 3d shape programs. 2019. 2, 3

[49] Trung-Thien Tran, Van-Toan Cao, and Denis Laurendeau. Extraction of reliable primitives from unorganized point clouds. *3D Research*, 2015. 2

[50] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-aware 3d model embedding and retrieval. In *European Conference on Computer Vision*, pages 397–413. Springer, 2020. 2

[51] Tamás Várady, Ralph R Martin, and Jordan Cox. Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, 29(4):255–268, 1997. Reverse Engineering of Geometric Models. 1, 2

[52] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges, 2020. 2

[53] Karl D. D. Willis, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *The 1st Workshop on Sketch-Oriented Deep Learning (SketchDL), CVPR 2021*, June 2021. 2

[54] 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 reconstruction. *arXiv preprint arXiv:2010.02392*, 2020. 2, 6

[55] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. 1, 2, 3, 6, 8

[56] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D. D. Willis, and Daniel Ritchie. Inferring cad modeling sequences using zone graphs. In *CVPR*, 2021. 2