# CAD-SIGNet: CAD Language Inference from Point Clouds using Layer-wise Sketch Instance Guided Attention

Mohammad Sadil Khan[†]
mdsadilkhan99@gmail.com

Elona Dupont[†]
elona.dupont@uni.lu

Sk Aziz Ali[†*]
sk_aziz.ali@dfki.de

Kseniya Cherenkova[‡†]
kseniya.cherenkova@uni.lu

Anis Kacem[†]
anis.kacem@uni.lu

Djamila Aouada[†]
djamila.aouada@uni.lu

[†]SnT, University of Luxembourg, [*]German Research Center for Artificial Intelligence, [‡]Artec3D

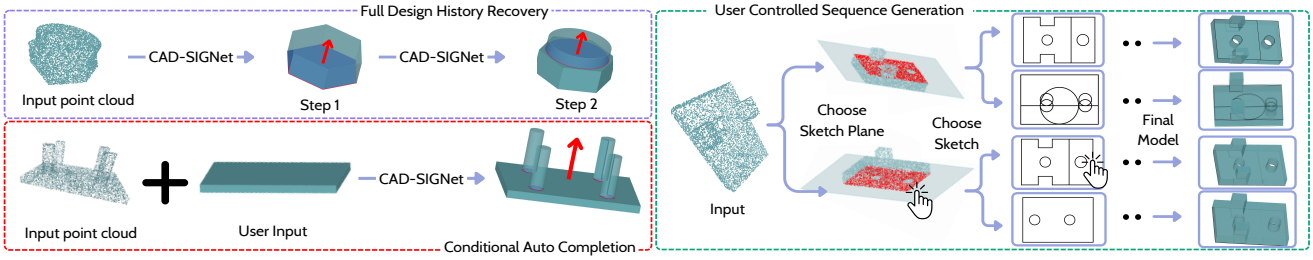


Figure 1. Full design history recovery from an input point cloud (top-left) and CAD-SIGNet - user interaction (bottom-left and right).

## Abstract

*Reverse engineering in the realm of Computer-Aided Design (CAD) has been a longstanding aspiration, though not yet entirely realized. Its primary aim is to uncover the CAD process behind a physical object given its 3D scan. We propose CAD-SIGNet, an end-to-end trainable and aeto-regressive architecture to recover the design history of a CAD model represented as a sequence of sketch-and-extrusion from an input point cloud. Our model learns CAD visual-language representations by layer-wise cross-attention between point cloud and CAD language embedding. In particular, a new Sketch instance Guided Attention (SGA) module is proposed in order to reconstruct the fine-grained details of the sketches. Thanks to its auto-regressive nature, CAD-SIGNet not only reconstructs a unique full design history of the corresponding CAD model given an input point cloud but also provides multiple plausible design choices. This allows for an interactive reverse engineering scenario by providing designers with multiple next step choices along with the design process. Extensive experiments on publicly available CAD datasets showcase the effectiveness of our approach against existing baseline models in two settings, namely, full design history recovery and conditional auto-completion from point clouds.*

## 1. Introduction

Computer-Aided Design (CAD) has become the *de facto* method for designing, drafting, and modeling in various industries [8, 32]. 3D reverse engineering is the process of inferring a CAD model given a 3D scan. This procedure requires the expertise of designers and can be time-consuming [9, 43]. Towards the automation of this procedure, several works focused on decomposing point clouds into parametric primitives allowing the reconstruction of the final CAD model [9, 16, 23, 25, 35]. However, CAD modeling consists of a sequential process where designers draw 2D sketches (*e.g.* lines, arcs) and apply CAD operations (*e.g.* extrusion, chamfer) [42, 43]. Recovering these intermediate design steps is crucial as it enables the editablity and re-usability of different object parts sharing the same functionality. For instance, a chair can be composed of three design steps, legs, seat, and back rest. Retrieving these steps can allow for editing the legs to be taller, reusing the back rest in another chair design, etc [19]. Nevertheless, identifying adequate design steps requires design expertise.

Mohammad Sadil Khan, Elona Dupont, Anis Kacem, and Djamila Aouada are affiliated to SnT, University of Luxembourg. Sk Aziz Ali is affiliated to SnT, University of Luxembourg and German Research Center for Artificial Intelligence. Most of the work conducted by Sk Aziz Ali took place when at SnT, University of Luxembourg. Kseniya Cherenkova is affiliated to both SnT, University of Luxembourg and Artec3D.

Accordingly, recent methods [31, 41, 42] attempted to learn this expertise from large-scale CAD datasets [41, 42]. In particular, the sequential nature of CAD modeling made language-like representations with adequate grammar an appealing choice [15, 42, 44]. While such a CAD language-like representation has been successfully adopted for CAD generative models [15, 33, 42, 44], it has not been established for 3D reverse engineering. As in point cloud captioning [4, 5], leveraging language-like representations for reverse engineering requires mechanisms for jointly learning visual representation from point clouds and corresponding CAD language. Hence, the main question that we ask is: ***how to effectively learn CAD visual-language representations from point cloud and CAD sequences for 3D reverse engineering?***

To answer this question, many challenges need to be addressed due to the structural disparity between point clouds in 3D space and language-like representations of CAD sequences [27]. In particular, CAD sequences encode both the chronological order of design steps and their parametric form [42, 44], while the corresponding point clouds only encode the geometry of the final design [9]. To the best of our knowledge, the only works that infer CAD language from point clouds are DeepCAD [42] and MultiCAD [27]. While DeepCAD [42] focused on learning CAD language using a feed-forward strategy and presented the point cloud to CAD language setting as a future application, MultiCAD [27] focused on learning the interaction of features from distinct modalities (i.e. point cloud and CAD language) through a contrastive learning framework. Despite their promising results, both methods suffer from two main limitations: (1) Both visual and CAD language representations are learned separately in the first stage. A mapping between the two representations is learned afterwards. Nevertheless, this separate learning might result in modality-specific features that are not relevant for CAD language inference from point clouds [36]; (2) the learning of CAD language representation is achieved using a feed-forward strategy where the CAD language of the full design history is inferred at once. However, in a real-world scenario, providing input or preferences at each design steps would allow for tailoring the solution to the requirements of the designer [44, 45].

To address the aforementioned challenges and limitations, we propose *CAD-SIGNet*, an end-to-end trainable architecture that auto-regressively infers CAD language in the form of *sketch-and-extrusion* design steps from point clouds. Instead of learning separate representations for both point clouds and CAD language and the mapping between them, the proposed method jointly learns these representations through multi-modal transformer blocks. Each block is composed of layer-wise cross-attention between CAD language and point cloud embedding. Moreover, other existing works [27, 42] infer sketches from a global representation of the point cloud. However, we assume that only a subset of the point cloud is needed to parameterize a sketch. As shown in the right panel of Figure 1, designers specify a plane in 3D space where the sketch is drawn. The intersection of the sketch region and the point cloud (shown in red in the same Figure) is assumed to be sufficient for sketch parameterization. Therefore, this subset, referred to as *Sketch Instance*, is first identified and then considered in the cross-attention to infer sketch parameters. We refer to this technique as *Sketch instance Guided Attention* (SGA). It allows the network to focus its attention on specific points (i.e. sketch instance), hence improving fine-grained sketch inference. Finally, the auto-regressive nature of CAD-SIGNet allows multiple plausible design choices to coexist. As shown in the right panel of Figure 1, this enables an interactive reverse engineering scenario, offering designers various choices throughout the CAD process. An overview of the proposed approach is provided in Figure 2.

**Contributions:** The contributions can be summarized to:

- An end-to-end trainable auto-regressive network that infers CAD language given an input point cloud. To the best of our knowledge, we are the first to propose an auto-regressive strategy for this problem.
- Multi-modal transformer blocks with a mechanism of layer-wise cross-attention between point cloud and CAD language embedding.
- A Sketch instance Guided Attention (SGA) module which guides the layer-wise cross-attention mechanism to attend on relevant regions of the point cloud for predicting sketch parameters.
- A thorough experimental validation in two different reverse engineering settings, namely, full CAD history recovery and conditional auto-completion from point clouds (see bottom left panel of Figure 1).

## 2. Related Works

**Deep Learning-based CAD Reverse Engineering:** CAD models are well defined 3D objects described by their geometric and topological properties. As such, some works address the reverse engineering problem by focusing on recovering the geometric features of CAD models from point clouds. This has been achieved using parametric fitting techniques either on the edges of the CAD model [7, 26, 28, 39] or on the surfaces [10, 12, 16, 23, 35, 46]. However, a parametric fitting approach can only provide information about the final CAD model and it lacks any insight into the design process and the intermediate steps that were used to create the CAD model. In order to address these limitations, another line of work [9, 13, 18, 34, 47, 48] models the CAD construction using *Constructive Solid Geometry* (CSG) [18]. CSG is a sequential method in CAD modeling that combines simple 3D shapes (e.g., cube, sphere) using boolean operations (e.g., union, intersection). While

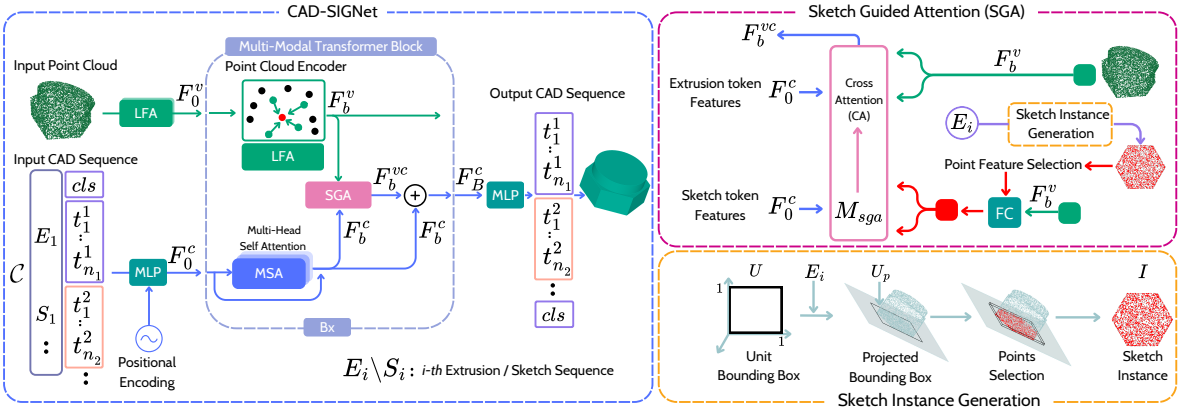


Figure 2. **Method Overview**. CAD-SIGNet (left) is composed of $\mathbf{B}$ Multi-Modal Transformer blocks, each consisting of an LFA [17] module to extract point features, $\mathbf{F}_b^v$, and a MSA [38] module for token features, $\mathbf{F}_b^c$. A SGA module (top right) combines $\mathbf{F}_b^v$ and $\mathbf{F}_b^c$ for CAD visual-language learning. A sketch instance (bottom right), $\mathbf{I}$, obtained from the predicted extrusion tokens is used to apply a mask, $\mathbf{M}_{\text{sga}}$ during CA to predict sketch tokens.

CSG can allow for the construction of relatively complex shapes, it is no longer the standard in the CAD industry [43]. Indeed, the *feature-based* approach has now been adopted by most CAD software as it allows for the modelling of more complex shapes using a sequence of sketches and CAD operations [41]. The work in [37] attempts to retrieve some of the features of the construction history as extrusion cylinders, but requires manual input to combine the cylinders into the final shape and does not result into parametric sketches. Self-supervised [24] and unsupervised [30] approaches have also been adopted in this context. Nevertheless, these approaches strive to infer plausible design steps approximating the input point cloud, but not necessarily inferring the standard parametric entities and therefore not reproducing design expertise. CAD-SIGNet goes beyond these limitations and leverages feature-based sequences of real design steps to predict CAD history from point clouds.

**CAD as a Language:** Due to the sequential nature of feature-based CAD modeling, a common strategy to represent it is to use language modelling. Inspired by Natural Language Processing (NLP) [38], some works have focused on language modeling of CAD sketches [15, 22, 33], others leveraged it in the context of CAD models [44, 45]. However, all the aforementioned works present generative models that allow for the manipulation of a latent space but do not directly tackle the reverse engineering problem. CAD-Parser [49] used an intermediate representation of the final shape, called *Boundary-Representation* (B-Rep) [21], instead of point cloud to relax the problem of CAD language inference. Closest to our work are DeepCAD [42] and MultiCAD [27]. DeepCAD proposed a language-based sketch-extrusion formulation and predicted the CAD history from point clouds as a preliminary experiment. Building on these findings, MultiCAD [27] opted for a two-stage multimodal contrastive learning strategy. In addition to the separate modality learning, both [42] and [27] use a feed-forward strategy limiting the scope of reverse engineering scenarios. In contrast, CAD-SIGNet presents a joint visual-language learning strategy and allows designers to interact with design choices (see Figure 1).

## 3. Problem and CAD Language Formulation

Given an input point cloud, our objective is to generate a sequence of tokens representing the design history of the corresponding CAD model. Formally, let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{N\times 3}$ be an input point cloud with $\mathbf{x}_i \in \mathbb{R}^3$ denoting the 3D coordinates of the $i$-th point and $N$ the number of points. Following recent CAD generative models [42, 44], the design history of a CAD model $\boldsymbol{\mathcal{C}} = \{\boldsymbol{\mathcal{C}}_j\}_{j=1}^{n_s}$ is represented by a sequence of $n_s$ design steps, where each step $\boldsymbol{\mathcal{C}}_j = \{t_k\}_{k=1}^{n_j}$ consists of a sequence of $n_j$ tokens $t_k \in [\![0..d_t]\!]$, with $d_t$ defining the tokenization interval. The objective is to learn a mapping,

$$\boldsymbol{\Phi} : \mathbb{R}^{N\times 3} \rightarrow [\![0..d_t]\!]^{n_{ts}} \quad \text{s.t.,} \quad \boldsymbol{\Phi}(\mathbf{X}) = \boldsymbol{\mathcal{C}} \,,$$

where $n_{ts} = \sum_{j=1}^{n_s} n_j$ denotes the total number of tokens. As in [42, 44], the design history is assumed to be composed of *sketch-and-extrusion* sequences. This implies that the sequence of tokens $\{t_k\}_{k=1}^{n_j}$ of each design step $\boldsymbol{\mathcal{C}}_j$ represents either a parametric sketch $\mathbf{S}$ or an extrusion operation $\mathbf{E}$ and the full design history $\boldsymbol{\mathcal{C}}$ can be seen as a sequence of sketch-and-extrusion pairs $\{(\mathbf{S}_l, \mathbf{E}_l)\}_{l=1}^{n_s/2}$.

**Sketch Representation:** Similarly to [44], a hierarchical representation of the sketch is considered. As depicted in Figure 3, a sketch is created from one or more faces, with a face being a 2D region bounded by loops. A loop, in turn, is a closed path that can consist of either a single closed curve, such as a circle, or multiple curves, e.g., combination of lines and arcs. The curves are represented by the tokenized 2D coordinates $(p_x, p_y)$ of their parametric formulation (e.g., start and end points for lines). The end of

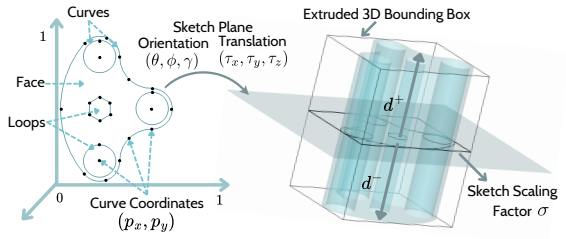


Figure 3. Illustration of sketch and extrusion representations.

a curve, loop, face, and sketch are represented by the end tokens $e_c$, $e_l$, $e_f$, and $e_s$, respectively.

**Extrusion Representation:** The extrusion operation defines the sketch plane and the parameters needed to turn it into a 3D volume. Following [44], the tokens $(\theta, \phi, \gamma)$ and $(\tau_x, \tau_y, \tau_z)$ define the sketch plane orientation and translation, respectively, with respect to a reference coordinate system. The token $\sigma$ scales the normalized sketch defined by the sketch tokens. The pair $(d^+, d^-)$ represents the extrusion distances along the normal direction of the sketch plane and its opposite, respectively. The parameter $\beta$ denotes the type of extrusion operation among *new*, *cut*, *join*, and *intersect*. Finally, $e_e$ sets the end of the extrusion tokens. Figure 3 shows the different tokens used to represent the extrusion operation.

In addition to sketch and extrusion tokens, $pad$ is used for padding and $cls$ is considered to indicate the start or end of the design sequence. More details about CAD sequence representation are provided in supplementary materials.

## 4. CAD-SIGNet Architecture

The proposed CAD-SIGNet is an end-to-end trainable transformer-based architecture that takes a point cloud $\mathbf{X}$ as an input and outputs the corresponding design history sequence $\boldsymbol{\mathcal{C}}$. It follows an auto-regressive strategy by considering the set of previous tokens $\boldsymbol{\mathcal{C}}_{<i} = \{t_j\}_{j<i}$ as context to infer the next token $t_i$. For a given point cloud $\mathbf{X}$, the goal of CAD-SIGNet is to learn its corresponding CAD history using the following probability distribution,

$$p_\theta(\boldsymbol{\mathcal{C}} \mid \mathbf{X}) = \prod_{i=1}^{n_{ts}} p_\theta(t_i \mid \{t_j\}_{j<i}, \mathbf{X}) \,, \tag{1}$$

where $t_i$ is the $i$-th sequence token and $\theta$ denotes the learned parameters of the network. As mentioned in Section 3, the predicted tokens $t_i$ correspond to the representations of sketch-and-extrusion sequences. Unlike other CAD language generative models [42, 44] which infer sketch tokens $\mathbf{S}_k$ for each design step $\boldsymbol{\mathcal{C}}_k$ followed by extrusion tokens $\mathbf{E}_{k+1}$, CAD-SIGNet first predicts extrusion tokens that are further used as context to predict sketch tokens. An overview of our CAD-SIGNet modules is provided in the left panel of Figure 2.

### 4.1. Point Cloud and CAD Language Embedding

The first module of CAD-SIGNet is responsible for embedding point cloud points and CAD language tokens into the same $d_e$-dimensional space $\mathbb{R}^{d_e}$.

**Point Cloud Embedding:** Given the point cloud $\mathbf{X} \in \mathbb{R}^{N\times(3+f)}$, where $f$ is the number of additional per-point estimated features[1], a linear layer[2] followed by ReLU [14] is firstly applied as follows,

$$\mathbf{F}_0^p = \mathrm{ReLU}(\mathbf{X}\mathbf{W}_{\text{emb}}^p) \,, \tag{2}$$

where $\mathbf{F}_0^p \in \mathbb{R}^{N\times d_e^{p_0}}$ is the learned embedding, $\mathbf{W}_{\text{emb}}^p \in \mathbb{R}^{(3+f)\times d_e^{p_0}}$ is a learnable matrix, and $d_e^{p_0} = 16$. The per-point features obtained in $\mathbf{F}_0^p$ are further enriched using two *Local Feature Aggregation* (LFA) [17] modules. LFA uses k-Nearest Neighbor (k-NN) to aggregate the features of neighboring points through a linear combination weighted by learned attention weights. A linear layer is applied on the resulting aggregated features followed by ReLU for each LFA module. The first LFA module results in the point cloud embedding $\mathbf{F}_0^v \in \mathbb{R}^{N\times d_e}$ defined by,

$$\mathbf{F}_0^v = \mathrm{ReLU}(\mathrm{LFA}(\mathbf{F}_0^p)\mathbf{W}_{\text{lfa}}) \,, \tag{3}$$

where $\mathbf{W}_{\text{lfa}} \in \mathbb{R}^{d_e^{p_0}\times d_e}$ denotes the weight matrix of the linear projection. The second LFA module is applied on $\mathbf{F}_0^v$ without changing its dimension. For more details about the operator $\mathrm{LFA}(.)$, readers are referred to [17].

**CAD Language Embedding:** Given an input design sequence $\boldsymbol{\mathcal{C}} = \{t_i\}_{i=1}^{n_{ts}} \in [\![0..d_t]\!]^{n_{ts}}$, a matrix form of the sequence is adopted. Unlike [44] which maps the sketch coordinates $p_x$ and $p_y$ into a 1-dimensional token, we consider them as a single 2-dimensional token $(p_x, p_y)$. To avoid dimension mismatch, the other tokens are also considered as 2-dimensional by augmenting them with $pad$ tokens. By concatenating these tokens and using a one-hot encoding, a matrix form $\mathbf{C} \in \{0,1\}^{n_{ts}\times 2d_t}$ is used to represent the sequence $\boldsymbol{\mathcal{C}}$. As in [44], token flags $\mathbf{C}_{\text{type}} \in [\![0..n_f]\!]^{n_{ts}\times 1}$ and $\mathbf{C}_{\text{step}} \in [\![0..n_s/2]\!]^{n_{ts}\times 1}$ are set to indicate token types and design steps, respectively. The initial embedding of the CAD language $\mathbf{F}_0^c \in \mathbb{R}^{n_{ts}\times d_e}$ is obtained by using the aforementioned token representations within a linear layer and is given by,

$$\mathbf{F}_0^c = [\mathbf{C} + \mathbf{M}_{\text{seq}}, \mathbf{C}_{\text{type}}, \mathbf{C}_{\text{step}}]\mathbf{W}_{\text{emb}}^c + \mathbf{C}_{\text{pos}} \,, \tag{4}$$

where $(,)$ is the concatenation operation, $\mathbf{W}_{\text{emb}}^c \in \mathbb{R}^{(2d_t+2)\times d_e}$ is a learnable weight matrix, and $\mathbf{C}_{\text{pos}} \in \mathbb{R}^{n_{ts}\times d_e}$ a learned positional encoding. Note that CAD sequences have a variable number of tokens $\tilde{n}_{ts} < n_{ts}$ and $\mathbf{M}_{\text{seq}} \in \{0, -\infty\}^{n_{ts}\times 2d_t}$ is the padding mask that sets token embedding beyond $\tilde{n}_{ts}$ to $-\infty$.

[1] Point normals are extracted using Open3D [1]

[2] All linear layers used in the paper consist of a weight matrix and a bias. For notation simplicity, we omit the bias.

### 4.2. Layer-wise Multi-Modal Transformer Block

Based on the aforementioned embedding, CAD-SIGNet jointly learns visual-language representations using $B$ multi-modal transformer blocks of layer-wise cross-attention between CAD and point cloud embedding.

In particular, let the CAD language embedding $\mathbf{F}^c_{b-1}$ and the point cloud embedding $\mathbf{F}^v_{b-1}$ be the input of the $b$-th block (i.e., the first block receives $\mathbf{F}^c_0$ defined in Eq. (4) and $\mathbf{F}^v_0$ defined in Eq. (3)). Firstly, $\mathbf{F}^c_b$ is generated from $\mathbf{F}^c_{b-1}$ using a multi-head scaled dot-product attention [38] (SA) and an add-normalization layer [38] (AddNorm) as follows

$$\mathbf{F}^c_b = \mathrm{SA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{M}) \,, \tag{5}$$

$$\mathbf{F}^c_b = \mathrm{AddNorm}(\mathbf{F}^c_b, \mathbf{F}^c_{b-1}) \,, \tag{6}$$

where query $\mathbf{Q}$, key $\mathbf{K}$, and value $\mathbf{V}$ are extracted from $\mathbf{F}^c_{b-1}$ and $\mathbf{M}$ is the standard self-attention mask [38]. On the other hand, the point cloud embedding $\mathbf{F}^v_{b-1}$ undergoes an additional LFA module as described in Eq. (3) to obtain a point cloud embedding $\mathbf{F}^v_b \in \mathbb{R}^{N \times d_e}$.

To enable the information passing between CAD language and point cloud embedding within each block, a cross-attention layer is used on $\mathbf{F}^v_b$ and $\mathbf{F}^c_b$. This is achieved by employing linear projections to extract a key $\mathbf{K}_v \in \mathbb{R}^{N \times d_e}$ and value $\mathbf{V}_v \in \mathbb{R}^{N \times d_e}$ from the point cloud embedding $\mathbf{F}^v_b$. A query $\mathbf{Q}_c \in \mathbb{R}^{n_{ts} \times d_e}$ is extracted from the CAD embedding $\mathbf{F}^c_b$. Using Eq. (5), the cross-attention layer computes a CAD visual-language embedding $\mathbf{F}^{vc}_b \in \mathbb{R}^{n_{ts} \times d_e}$ as follows,

$$\mathbf{F}^{vc}_b = \mathrm{SA}(\mathbf{Q}_c, \mathbf{K}_v, \mathbf{V}_v, \mathbf{0}) \,, \tag{7}$$

where $\mathbf{0}$ is a $(n_{ts} \times N)$ zero matrix. Furthermore, the cross and self-attended embedding $\mathbf{F}^{vc}_b$ and $\mathbf{F}^c_b$ are added and normalized to help the network learn the geometric relationship between CAD tokens, yielding, $\mathbf{F}^c_b = \mathrm{AddNorm}(\mathbf{F}^{vc}_b, \mathbf{F}^c_b)$. Finally, as in [38], a Feed-Forward Network (FFN) is applied on $\mathbf{F}^c_b$ and added to it to form the final CAD embedding, which is passed to the next block along with $\mathbf{F}^v_b$.

**Sketch Instance Guided Attention (SGA):** The aforementioned multi-modal transformer blocks are designed to pass the information from all point embedding to CAD token embedding. However, we posit that parameterizing a sketch requires only cross-attending to a subset of the point cloud. As shown in the bottom right panel of Figure 2, the intersection between the sketch region and the point cloud (depicted in red in the same Figure) is considered as adequate for sketch parameterization. As depicted in Figure 3, the representation of extrusion tokens defines the sketch plane and bounding box. Furthermore, CAD-SIGNet predicts extrusion tokens followed by sketch tokens for each design step. This implies that the predicted extrusion tokens can be leveraged to define a *sketch instance* on the point cloud for cross-attention with sketch token embedding.

**Definition 1** *A sketch instance* $\mathbf{I} \in \mathbb{R}^{\eta \times 3} \subset \mathbf{X}$*, with* $\eta < N$*, is a subset of the input point cloud* $\mathbf{X}$*. It is extracted by selecting points inside the bounding box on the sketch plane derived from the corresponding predicted extrusion tokens.*

The bottom right panel of Figure 2 shows the sketch instance extraction process. Given a set of extrusion tokens $\mathbf{E}$, we first project the unit bounding box of the $xy$-plane into a bounding box on the sketch plane defined by the extrusion tokens $\mathbf{E}$. In particular, given the unit bounding on $xy$-plane defined by the points $\mathbf{U} = [(0,0,0)^\mathbf{T}, (0,1,0)^\mathbf{T}, (1,0,0)^\mathbf{T}] \in \mathbb{R}^{3\times3}$, the Euler angles $(\theta, \phi, \gamma)$, the translation vector $(\tau_x, \tau_y, \tau_z)$, and the scaling factor $\sigma$ defined by the extrusion operation $\mathbf{E}$, the projected bounding box $\mathbf{U}_p \in \mathbb{R}^{3\times3}$ is given by,

$$\mathbf{U}_p = (\mathbf{R}_{xyz}(\theta, \phi, \gamma)(\mathbf{U} \times \sigma) + (\tau_x, \tau_y, \tau_z)^\mathbf{T}) \,, \tag{8}$$

where $\mathbf{R}_{xyz}(\theta, \phi, \gamma) \in \mathcal{SO}(3)$ combines the Euler angles in a rotation matrix in the special orthogonal group $\mathcal{SO}(3)$. The sketch instance $\mathbf{I}$ is then defined by the points of $\mathbf{X}$ lying inside this bounding box, i.e., $\mathbf{I} := \{\mathbf{x} \in \mathbf{X} \mid \phi(\mathbf{x}, \mathbf{U}_p) = \text{True}\}$, where $\phi(\mathbf{x}, \mathbf{U}_p)$ is an operator that checks whether an input point $\mathbf{x} \in \mathbb{R}^3$ is inside the projected bounding box $\mathbf{U}_p$. Note that for training the ground-truth extrusion tokens are used to define the bounding box $\mathbf{U}_p$, while the predicted extrusion tokens are leveraged at inference time. In order to not penalize small errors in sketch plane predictions of the extrusion tokens and point cloud sampling, the bounding box is enlarged in the direction of sketch plane normal and its opposite by a small margin $0.1 \times \max(d^+, d^-)$. The extracted sketch instances can be then used in the cross-attention defined in Eq. (7) only for sketch token embedding by employing a suitable mask instead of the zero matrix. In particular, let $\mathbf{M}_{\text{sga}} \in \{0, -\infty\}^{n_{ts} \times N}$ be this mask and $m^{\text{sga}}_{ij}$ be its value for the attention between the $i$-th token and $j$-th point embedding. $\mathbf{M}_{\text{sga}}$ is introduced to mask the attention of sketch token embedding to the points lying outside their corresponding sketch instance. As a result, $m^{\text{sga}}_{ij}$ is set to $0$ if the $i$-th token embedding is not denoting a sketch. If the $i$-th token is representing a sketch, then $m^{\text{sga}}_{ij}$ is set to $0$ where the $j$-th point embedding is part of the corresponding sketch instance and $-\infty$ otherwise. Note that after identifying the sketch instances, $4$ linear layers are used on the corresponding subsets of $\mathbf{F}^v_b$ to refine their embedding before extracting the key and value for the cross-attention with sketch token embedding. The top right panel of Figure 2 visually describes the SGA module.

### 4.3. Training and Inference Strategies

After the last multi-modal transformer block, the CAD embedding $\mathbf{F}^c_B$ is passed to two separate linear lay-

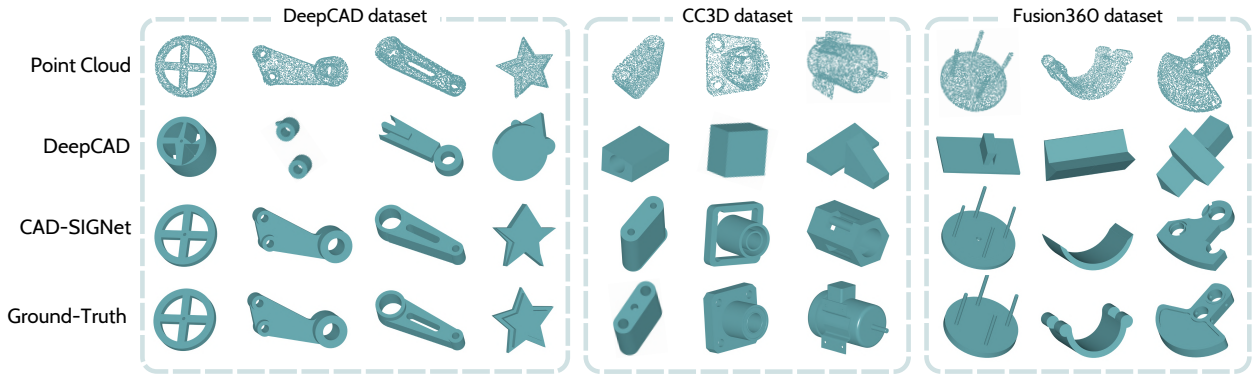


Figure 4. Visual results of reconstruction from the CAD sequences predicted from input point clouds. Both DeepCAD [42] and CAD-SIGNet are trained on DeepCAD dataset [42]. **Left**: Results on DeepCAD dataset [42]. **Middle**: Cross-dataset results on CC3D dataset [6], **Right**: Cross-dataset results on Fusion360 dataset [41].

ers for predicting the 2D tokens probability matrices $\mathbf{O}_x$, $\mathbf{O}_y \in [0,1]^{n_{ts} \times d_t}$.

**Training:** During training, a teacher-forcing strategy [40] is used to pass the ground-truth as input. The cross-entropy loss $\mathcal{L}_{ce}$ is used as an objective function.

**Inference:** During inference, given the input point cloud $\mathbf{X}$ and the initial CAD sequence consisting of $\mathcal{C} = \{(cls, pad)\}$, the next tokens are auto-regressively generated until the *end* token is predicted.

**Hybrid Sampling:** The auto-regressive nature of CAD-SIGNet suggests that different token predictions at a given time-stamp result in different final CAD sequences. This allows for generating multiple plausible predictions given a point cloud. In particular, given the output probabilities $\mathbf{O}_x, \mathbf{O}_y$, one can take the *top*-1 to obtain the predicted tokens or opt for a different selection strategy for each token to have a different final CAD sequence. To showcase this, we use a hybrid sampling approach during inference by selecting *top*-5 probabilities for the first token, and *top*-1 for subsequent tokens. This results in 5 different final CAD sequences given a point cloud. Finally, the optimal CAD sequence is chosen by selecting the one that best approximates the input point cloud. This is assessed by reconstructing the CAD models[3] from the predicted sequences, sampling point clouds on them, and selecting the model that results in a minimum Chamfer Distance [11] with respect to the input point cloud.

## 5. Experimental Results

In this section, the proposed CAD-SIGNet is evaluated on two reverse engineering scenarios: (1) design history recovery from point clouds, and (2) conditional auto-completion of design history given user input and point clouds. Additional preliminary experiments showcasing the applicability of the proposed method in a realistic scenario of reverse engineering is also discussed.

**Dataset**: The DeepCAD dataset [42] is used. The sketch and extrusion parameters are normalized, ensuring that the sketches and the CAD models are within a unit-bounding box starting from the origin. The point clouds are obtained by uniformly sampling 8192 points from the normalized CAD model. As in [42], the sketch and extrusion parameters are quantized to 8 bits.

**Implementation Details**: We use 8 CAD-SIGNet multimodal transformer blocks with $h = 8$ number of heads for self-attention. The latent dimension is set to $d_e = 128$. The network has been trained with a batch size of 72 for 150 epochs using 2 *NVIDIA RTX A6000* GPUs. We implement curriculum learning [3] for the first 15 epochs, increasingly sorting CAD sequences by the number of curves. For the subsequent 135 epochs, the samples are shuffled. More details are provided in supplementary materials.

### 5.1. Design History Recovery from Point Cloud

In this experiment, the task is to infer CAD language history given an input point cloud.

**Metrics**: To evaluate the predicted sequences, a set of metrics assessing different levels of the predictions are used. In particular, the final CAD reconstructions are quantitatively evaluated with respect to ground-truth CAD models using mean and median *Chamfer Distances* (CD) [11]. As CAD sequences are predicted as tokens, they do not necessarily result in valid CAD models when reconstructed using OpenCascade [2]. Accordingly, an *Invalidity Ratio* (IR) metric, expressed as a percentage, is the ratio of invalid models. *F1* score is computed to evaluate the predicted extrusions and different primitive types along with their occurrences in the sequences. A Hungarian matching algorithm [20] is used to match the predicted loop and primitive bounding boxes with the ground-truth ones of the same sketch. More details are provided in the supplementary.

[3] Opencascade[2] is used to reconstruct a model from a CAD sequence.

| Model | IR↓ | Mean CD ↓ ($\times 10^{-3}$) | Median CD ↓ ($\times 10^{-3}$) |
|---|---|---|---|
| DeepCAD [42] | 7.14 | 42.49 | 9.640 |
| MultiCAD [27] | 11.5 | - | 8.090 |
| **CAD-SIGNet (Ours)** | **0.88** | **3.430** | **0.283** |

Table 1. Design history recovery from point clouds on DeepCAD [42] dataset. Invalidity Ratio (IR), mean and median Chamfer Distance (CD) results.

**Results:** To the best of our knowledge, DeepCAD and MultiCAD are the only works in literature that perform point cloud to CAD language inference. Note that DeepCAD results have been reproduced using its publicly available implementation, while MultiCAD results were taken from the reported ones in [27] due to unavailability of public implementation. It can be observed in Table 1 that the proposed CAD-SIGNet is outperforming both DeepCAD and MultiCAD in terms of median CD by a factor of 35 and 28, respectively. Moreover, the IR metric shows that the predicted CAD sequences by CAD-SIGNet results in drastically more valid CAD model reconstructions than both DeepCAD and MultiCAD. In Table 2, the per-primitive and extrusion F1 scores of our method are compared to those of DeepCAD. Our model predicts more accurately the primitive types, and their occurrences in the design sequence when compared to DeepCAD. Notably, our method records improvements of more than 14% in F1 score on the arc type with respect to DeepCAD. In addition, CAD-SIGNet correctly predicts the extrusions in most cases, showing that our model can correctly identify the number of needed design steps. Figure 4 displays several qualitative CAD models reconstructed from the predicted CAD sequences. Visually, our method achieves better reconstructions with more accurate details than DeepCAD [42]. More visual results are reported in the supplementary materials.

| Model | F1 Score | | | |
|---|---|---|---|---|
| | Line ↑ | Arc↑ | Cricle↑ | Extrusion↑ |
| DeepCAD[42] | 69.26 | 13.82 | 60.14 | 86.70 |
| **CAD-SIGNet (Ours)** | **77.31** | **28.65** | **70.36** | **92.72** |

Table 2. Design history recovery from point clouds on DeepCAD [42] dataset. F1 scores for lines, arcs, circles, and extrusions.

**Ablation Study:** The impact of the components proposed in CAD-SIGNet is assessed in Table 3 in terms of F1 scores and CAD reconstruction metrics (IR, mean, and median CD). In the first row of this Table, the hybrid sampling described in Section 4.3 is ablated, thus selecting tokens with *top-1* probabilities. It can be observed that this results in a drop of performance in terms of CD distances and IR while maintaining similar sketch and extrusion scores. A similar trend is observed for the second row, where the SGA module is omitted in the layer-wise cross-attention described in Section 4.2. This suggests that the hybrid sampling and the SGA module are especially important to obtain valid and accurate final CAD reconstructions. Finally,

| Model | F1 Score | | Median CD↓ ($\times 10^{-3}$) | Mean CD↓ ($\times 10^{-3}$) | IR↓ |
|---|---|---|---|---|---|
| | Sketch↑ | Extrusion↑ | | | |
| Ours w/o Hybrid Samp. | 75.30 | **92.97** | 0.291 | 6.784 | 5.02 |
| Ours w/o SGA | 75.13 | 92.49 | 0.289 | 4.995 | 2.18 |
| Ours w/o Layer-wise CA | 45.89 | 84.53 | 76.40 | 122.7 | **0** |
| **CAD-SIGNet (Ours)** | **75.94** | 92.72 | **0.283** | **3.432** | 0.88 |

Table 3. Ablation study. Sketch and extrusion F1 scores. Median, Mean CD, and IR metrics.

the third row reports the results when the layer-wise cross-attention defined in Eq. (7) is not considered. In this case, each CAD language embedding $\mathbf{F}_b^c$ cross-attends to only the point cloud embedding $\mathbf{F}_B^v$ from the last block $B$. In other words, this experiment omits passing the information from intermediate geometric embedding to the CAD language one. Despite generating only valid CAD reconstructions, we observe a drastic performance drop in all other metrics using this strategy compared to the proposed layer-wise cross-attention. Visual results of the ablation experiments are provided in supplementary materials.

## 5.2. Conditional Auto-Completion from User Input

CAD-SIGNet's auto-regressive nature enables it to complete the next design steps given a user input and a complete point cloud. To showcase this scenario, the same model trained for full design history recovery is used. During inference, the ground-truth tokens of the first extrusion and sketch are provided, and the task is to predict the next tokens of the CAD sequence given the complete point cloud.

**Baseline Methods:** To the best of our knowledge, there is no existing method capable of achieving the aforementioned task. DeepCAD [42] and MultiCAD [27] are not suitable candidates for this task due to their feed-forward nature. For the sake of comparison, we adapt two auto-regressive generative models, namely SkexGen [44], and HNC [45]. Similarly to DeepCAD [42], we train a PointNet++ [29] to encode the point cloud into the latent space learned by SkexGen [44], and HNC [45] on CAD language. Note that these adapted baselines were not subject to optimization. More details on how these methods are adapted are provided in the supplementary materials.

**Metrics:** The auto-completion performance is evaluated in terms of final CAD reconstructions. This is measured by the IR introduced in Section 5.1 and another measure based on the CD. The latter is given by computing: (1) $\text{CD}^{\text{gt}}_{\text{pred}}$ which is the CD between the CAD reconstruction of the completed sequence and the ground-truth CAD model, (2) $\text{CD}^{\text{gt}}_{\text{in}}$ which is the CD between the CAD reconstruction of the user input sequence and the ground-truth, (3) the ratio of the two distances $\text{CD}^{\text{gt}}_{\text{pred}}/\text{CD}^{\text{gt}}_{\text{in}}$. This ratio allows for assessing whether the completed sequence resulted in a better final CAD reconstruction with respect to the user input. In order to reflect the distribution of this measure, the three quartiles Q1, Q2 (i.e., median), and Q3 are reported.

**Results:** Table 4 compares the results of CAD-SIGNet

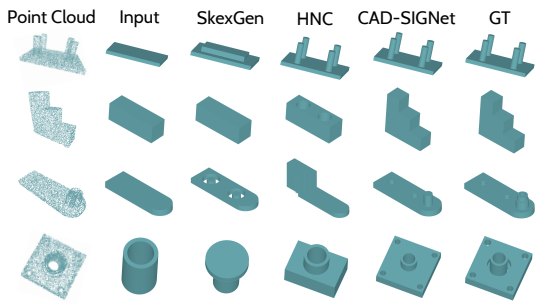


Figure 5. Visual results for auto-completion from user input on DeepCAD [42] dataset. From left to right, input point cloud, CAD model reconstruction from user input CAD sequence, SkexGen [44], HNC [45], CAD-SIGNet (ours), and ground-truth.

when using hybrid sampling and without, to the adapted baselines based on HNC [45] and SkexGen [44]. It can be observed that all the quartile values of the CD ratio for SkexGen baseline are very close to 1 which indicates that the completed sequences resulted in a final CAD reconstruction that is close to the one of the user input. Notably, CAD-SIGNet achieved low Q1 and Q2 values of 0.054 and 0.325, respectively, showing that it improved the user input by a large margin on half of the testing samples. These observations are consistent with the visual results reported in Figure 5. Moreover, discarding the hybrid sampling yields in a lower performance in all metrics but still outperforms both HNC and SkexGen baselines. Finally, the SGA module provides a noticeable improvement on the median CD ratio which further highlights its importance.

| Model | CD Ratio Q1↓ | CD Ratio Q2 (Median)↓ | CD Ratio Q3↓ | IR↓ |
|---|---|---|---|---|
| SkexGen-Baseline [44] | 0.987 | 1.000 | 1.035 | 2.04 |
| HNC-Baseline [45] | 0.437 | 1.015 | 2.589 | 8.85 |
| Ours w/o Hybrid Samp. | 0.096 | 0.696 | 1.096 | 4.40 |
| Ours w/o SGA | 0.060 | 0.458 | **0.992** | 0.91 |
| **CAD-SIGNet (Ours)** | **0.054** | **0.325** | 0.995 | **0.65** |

Table 4. Conditional auto-completion from user input and point clouds on DeepCAD [42] dataset. Quartiles of CD ratio and IR.

### 5.3. Applications of CAD-SIGNet

In this section, we highilight the applicability of CAD-SIGNet in a real-world scenario of reverse engineering. **Cross-Dataset Experiment on Fusion360**: Following the protocol outlined in MultiCAD [27], a cross-dataset experiment is performed on the Fusion360 dataset [41]. Results presented in Table 5 shows that CAD-SIGNet outperforms both MultiCAD [27] and DeepCAD [42] by a significant margin. Figure 4 (right) shows some visual comparison of the reconstructed CAD models from the predicted CAD sequences of CAD-SIGNet and DeepCAD [42]. The results indicate that CAD-SIGNet achieves better 3D reconstruction quality in comparison to DeepCAD [42].

**Design History Recovery from Realistic 3D Scans**: The reported results on DeepCAD dataset [42] are obtained by applying the model on point clouds sampled on CAD meshes. However, in a real-world scenario of reverse engineering, we aim to reverse engineer 3D scans which are prone to 3D scanning artifacts. The CAD-SIGNet model trained on DeepCAD dataset is tested on this setting by performing a cross-dataset testing. The CC3D dataset consists of 50k+ realistic 3D scans with their corresponding CAD models. Figure 4 shows some qualitative results of CAD-SIGNet compared to DeepCAD. Despite not being trained on such scan data, CAD-SIGNet succeeds in reconstructing promising CAD reconstructions. On the test set of CC3D dataset, composed of 5570 samples, we report a median CD of 2.398 and an IR of 2.39 compared to a median CD of 263.56 and an IR of 12.73 achieved by DeepCAD.

| Model | IR↓ | Median CD ($\times 10^{-3}$)↓ |
|---|---|---|
| DeepCAD [42] | 25.17 | 89.2 |
| MultiCAD [27] | 16.52 | 42.2 |
| **CAD-SIGNet (Ours)** | **1.83** | **1.15** |

Table 5. Cross-dataset experiment on design history recovery from point clouds on Fusion360 [41] dataset.

**User Controlled Reverse Engineering**: In a real-world reverse engineering scenario, it is not only desirable to generate the correct CAD sequence from a given point cloud, but to provide the user with a choice over every design step [45]. Towards this direction, one can further extend the hybrid sampling strategy to generate either multiple sketch planes for the extrusion steps or multiple sketches or loops from one single sketch plane. Since sketch sequence generation relies on the points laying close to the predicted sketch planes, changing sketch planes can result in a new sketch. The right panel of Figure 1 shows different generated design paths by our method a user can interactively follow.

## 6. Conclusion

In this paper, we propose, CAD-SIGNet, an auto-regressive architecture designed for recovering the design history of a CAD model given a point cloud. This history is represented as a sequence of sketch-and-extrusion sequences. Leveraging its auto-regressive nature, CAD-SIGNet reconstructs a CAD design history from the input point cloud, simultaneously offering a range of plausible design alternatives. Through multi-modal transformer blocks of layer-wise cross-attention, the information is passed between CAD language and point cloud embedding. Notably, the incorporation of the SGA module enhances the reconstruction of fine-grained details in the sketches. As future works, selecting subsets of points using SGA could help overcoming the high computational costs associated with large point clouds. While our work only considers extrusions, CAD-SIGNet could be adapted to other operations.

**Acknowledgement:** The present work is supported by the National Research Fund, Luxembourg under the BRIDGES2021/IS/16849599/FREE-3D and IF/17052459/CASCADES projects and Artec3D.

## References


[1] Open3d. http://www.open3d.org/, . 4

[2] Open cascade. https://dev.opencascade.org/, . 6

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. 6

[4] Daigang Cai, Lichen Zhao, Jing Zhang, Lu Sheng, and Dong Xu. 3djcg: A unified framework for joint dense captioning and visual grounding on 3d point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16464–16473, 2022. 2

[5] Zhenyu Chen, Ali Gholami, Matthias Nießner, and Angel X Chang. Scan2cap: Context-aware dense captioning in rgb-d scans. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 3193–3203, 2021. 2

[6] Kseniya Cherenkova, Djamila Aouada, and Gleb Gusev. Pvdeconv: Point-voxel deconvolution for autoencoding cad construction in 3d. pages 2741–2745, 2020. 6

[7] Kseniya Cherenkova, Elona Dupont, Anis Kacem, Ilya Arzhannikov, Gleb Gusev, and Djamila Aouada. Sepicnet: Sharp edges recovery by parametric inference of curves in 3d shapes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2023. 2

[8] Yuanzhe Deng, James Chen, and Alison Olechowski. What Sets Proficient and Expert Users Apart? Results of a Computer-Aided Design Experiment. Journal of Mechanical Design, 146(1):011401, 2023. 1

[9] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. Inversecsg: Automatic conversion of 3d models to csg trees. ACM Transactions on Graphics (TOG), 2018. 1, 2

[10] Eric-Tuan, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J Mitra. Cpfn: Cascaded primitive fitting networks for high-resolution point clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 7457–7466, 2021. 2

[11] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2463–2471, Los Alamitos, CA, USA, 2017. IEEE Computer Society. 6

[12] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981. 2

[13] Markus Friedrich, Pierre-Alain Fayolle, Thomas Gabor, and Claudia Linnhoff-Popien. Optimizing evolutionary csg tree extraction. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 1183–1191, 2019. 2

[14] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. Biological Cybernetics, 20:121–136, 1975. 4

[15] Y. Ganin, S. Bartunov, Y. Li, E. Keller, and S. Saliceti. Computer-aided design as language. Advances in Neural Information Processing Systems, 34, 2021. 2, 3

[16] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. ACM Transactions on Graphics (TOG), 2022. 1, 2

[17] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Learning semantic segmentation of large-scale point clouds with random sampling. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. 3, 4

[18] Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. Ucsg-net-unsupervised discovering of constructive solid geometry tree. Advances in Neural Information Processing Systems, 33:8776–8786, 2020. 2

[19] Milin Kodnongbua, Benjamin Jones, Maaz Bin Safeer Ahmad, Vladimir Kim, and Adriana Schulz. Reparamcad: Zero-shot cad re-parameterization for interactive manipulation. In SIGGRAPH Asia 2023 Conference Papers, pages 1–12, 2023. 1

[20] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly, 2(1–2):83–97, 1955. 6

[21] Joseph G Lambourne, Karl DD Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 12773–12782, 2021. 3

[22] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Free2cad: Parsing freehand drawings into cad commands. ACM Transactions on Graphics (TOG), 41(4):1–16, 2022. 3

[23] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, L. Yi, and Leonidas J. Guibas. Supervised fitting of geometric primitives to 3d point clouds. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2647–2655, 2018. 1, 2

[24] Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. 3

[25] Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. Surface and edge detection for primitive fitting of point clouds. In ACM SIGGRAPH 2023 Conference Proceedings, 2023. 1

[26] Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan Dirk Wegner. Pc2wf: 3d wireframe reconstruction from raw point clouds. arXiv preprint arXiv:2103.02766, 2021. 2

[27] Weijian Ma, Minyang Xu, Xueyang Li, and Xiangdong Zhou. Multicad: Contrastive representation learning for multi-modal 3d computer-aided design models. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, page 1766–1776,

New York, NY, USA, 2023. Association for Computing Machinery. 2, 3, 7, 8

[28] Dimitrios Mallis, Ali Sk Aziz, Elona Dupont, Kseniya Cherenkova, Ahmet Serdar Karadeniz, Mohammad Sadil Khan, Anis Kacem, Gleb Gusev, and Djamila Aouada. Sharp challenge 2023: Solving cad history and parameters recovery from point clouds and 3d scans. overview, datasets, metrics, and baselines. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1786–1795, 2023. 2

[29] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the 31st International Conference on Neural Information Processing Systems, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. 7

[30] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. In ECCV, 2022. 3

[31] Daniel Ritchie, Paul Guerrero, R Kenny Jones, Niloy J Mitra, Adriana Schulz, Karl DD Willis, and Jiajun Wu. Neurosymbolic models for computer graphics. In Computer Graphics Forum, pages 545–568. Wiley Online Library, 2023. 2

[32] D. Robertson and T.J. Allen. Cad system use and engineering performance. IEEE Transactions on Engineering Management, 40(3):274–282, 1993. 1

[33] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. In International Conference on Learning Representations, 2021. 2, 3

[34] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 2

[35] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16. Springer, 2020. 1, 2

[36] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In Proceedings of the IEEE/CVF international conference on computer vision, pages 7464–7473, 2019. 2

[37] 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 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11850–11860, 2022. 3

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. 3, 5

[39] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. Advances in neural information processing systems, 33:20167–20178, 2020. 2

[40] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. pages 270–280, 1989. 6

[41] Karl DD 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): 1–24, 2021. 2, 3, 6, 8

[42] 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), pages 6772–6782, 2021. 1, 2, 3, 4, 6, 7, 8

[43] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl DD Willis, and Daniel Ritchie. Inferring cad modeling sequences using zone graphs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6062–6070, 2021. 1, 3

[44] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In International Conference on Machine Learning (ICML), pages 24698–24724, 2022. 2, 3, 4, 7, 8

[45] Xiang Xu, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Karl D.D. Willis, and Yasutaka Furukawa. Hierarchical neural coding for controllable cad model generation. In Proceedings of the 40th International Conference on Machine Learning. JMLR.org, 2023. 2, 3, 7, 8

[46] Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qi-Xing Huang. Hpnet: Deep primitive segmentation using hybrid representations. 2021 ieee. In CVF International Conference on Computer Vision (ICCV)(2021), pages 2733–2742, 2021. 2

[47] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. Capri-net: Learning compact cad shapes with adaptive primitive assembly. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 2

[48] Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. D2csg: Unsupervised learning of compact csg trees with dual complements and dropouts. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. 2

[49] Shengdi Zhou, Tianyi Tang, and Bin Zhou. Cadparser: A learning approach of sequence modeling for b-rep cad. 3