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

Ahmet Serdar Karadeniz
SnT, University of Luxembourg
ahmet.karadeniz@uni.lu

Dimitrios Mallis
SnT, University of Luxembourg
dimitrios.mallis@uni.lu

Danila Rukhovich
SnT, University of Luxembourg
danila.rukhovich@uni.lu

Kseniya Cherenkova
SnT, University of Luxembourg, Artec 3D
kseniya.cherenkova@uni.lu

Anis Kacem
SnT, University of Luxembourg
anis.kacem@uni.lu

Djamila Aouada
SnT, University of Luxembourg
djamila.aouada@uni.lu

Abstract

Computer-Aided Design (CAD) plays a foundational role in modern manufacturing and product development, often requiring designers to modify or build upon existing models. Converting 3D scans into parametric CAD representations—a process known as CAD reverse engineering—remains a significant challenge due to the high precision and structural complexity of CAD models. Existing deep learning-based approaches typically fall into two categories: bottom-up, geometry-driven methods, which often fail to produce fully parametric outputs, and top-down strategies, which tend to overlook fine-grained geometric details. Moreover, current methods neglect an essential aspect of CAD modeling: sketch-level constraints. In this work, we introduce a novel approach to CAD reverse engineering inspired by how human designers manually perform the task. Our method leverages multi-plane cross-sections to extract 2D patterns and capture fine parametric details more effectively. It enables the reconstruction of detailed and editable CAD models, outperforming state-of-the-art methods and, for the first time, incorporating sketch constraints directly into the reconstruction process.

1 Introduction

Computer-Aided Design (CAD) modeling, typically performed using specialized software [1, 2, 3], plays a critical role in the development and manufacturing of real-world objects. In modern CAD workflows, designers begin by creating 2D sketches composed of parametric curves (e.g., lines, arcs, circles), which are further constrained by geometric relationships (e.g., coincident, tangent, parallel) [4]. These sketches are then transformed into 3D geometry using operations such as extrusion, revolution, and cutting. By following this sequential process, designers can construct 3D parametric solids that accurately represent the intended object. A common and practical approach in CAD workflows is to begin with a template of an existing object and adapt it to new design requirements. However, existing real-world objects commonly do not have publicly available CAD models. In such cases, a digital representation can be obtained through 3D scanning, which usually produces a 3D mesh. While meshes capture the surface geometry of an object, they are unstructured and lack the parametric information needed for further design and modification in CAD software. Consequently,

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

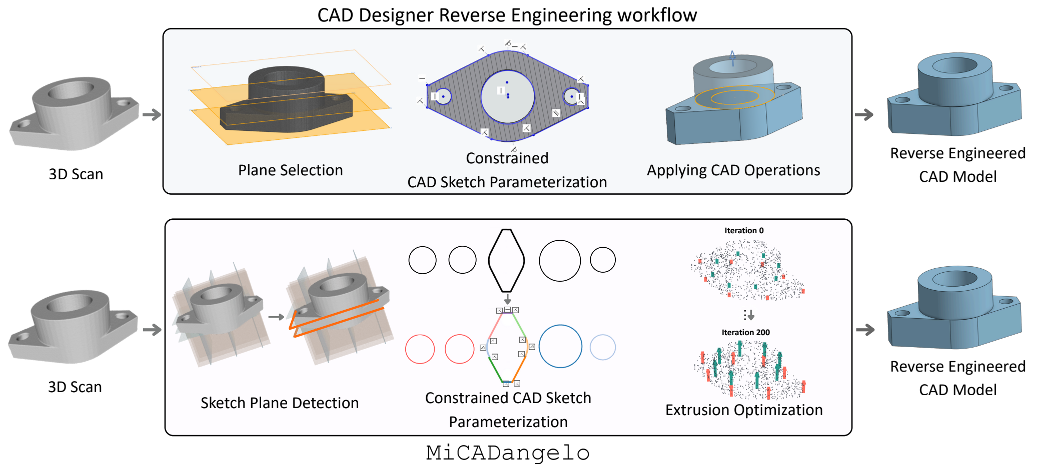

Figure 1: `MiCADangelo` is a novel framework for CAD reverse-engineering that mimics human design workflows. It analyzes 3D scans via 2D cross-sections to detect sketch planes, predict constrained parametric sketches and optimize extrusions.

converting a 3D mesh into a parametric CAD model—an operation commonly referred to as *CAD reverse engineering*—is a crucial step for enabling efficient design iteration and customization.

Today, CAD reverse engineering is typically carried out manually by designers using specialized software [5]. As illustrated in the top part of Figure 1, the process generally involves the following steps: (i) extracting a 2D cross-section of a specific part from the imported 3D scan; (ii) reconstructing the contour of the section using parametric 2D curves and placing the CAD constraints to restrict the modification; and (iii) applying CAD operations (e.g., extrusion) to the parametric sketch to create a 3D solid model of the part. These steps are repeated for the remaining parts until the entire 3D scan is reconstructed as a complete parametric CAD model within the software. While this process allows for faster CAD modeling compared to creating models from scratch, it remains tedious and demands CAD expertise. As a result, automating this process became an active research area [6, 7, 8, 9, 10, 11, 12].

Automated 3D CAD reverse engineering has advanced from identifying parts in 3D scans [13, 14], to fitting parametric primitives to scanned geometry [15, 16], and more recently, to predicting sequential design step directly from point clouds [17, 9]. The latter approach is particularly valuable, as it supports parametric reconstruction while preserving an editable design history. This advancement has been enabled by the emergence of datasets such as DeepCAD [6] and Fusion 360 [18], which provide sequences of 2D sketches and extrusion operations—commonly referred to as *sketch-extrude* sequences. Several recent methods have explored learning the mapping from point clouds to sketch-extrude sequences, which can be broadly categorized into: (i) top-down approaches [17, 9, 19], which directly predict the parameters of the sketch-extrude sequence as a series of parametric values from the input point cloud; and (ii) bottom-up approaches [8, 20, 7], which attempt to predict individual extrusion cylinders that, when combined, closely approximate the target geometry. While top-down approaches offer the advantage of producing fully parametric outputs that approximate ground-truth sketch-extrude sequences, they often struggle to capture fine-grained details of the input geometry frequently dominated by larger structures. In contrast, bottom-up approaches can better preserve local geometrical details, but they lack full parametrization and do not integrate seamlessly into standard CAD workflows [17]. Furthermore, to the best of our knowledge, none of these approaches consider CAD sketch constraints which are a critical component of CAD modeling [4, 21].

In this work, we propose `MiCADangelo`, a solution for automating CAD reverse engineering by emulating the way human designers approach the task. As illustrated in the bottom part of Figure.1, `MiCADangelo` begins by analyzing the input scan through a series of 2D cross-sectional slices and predicting key slices to serve as sketch planes. For each selected plane, closed loops are extracted from the cross-section and converted into raster images, which are used to predict both the 2D parametric curves and associated CAD sketch constraints. These constrained sketches are then extruded via an optimization process that leverages the local mesh geometry around each loop. The resulting extruded parts are ultimately merged to produce a structured, fully parametric CAD model.

`MiCADangelo` offers the advantage of preserving fine-grained geometric details of the input scan while generating fully parametric sketch-extrude sequences, including CAD constraints—an aspect largely unexplored in 3D CAD reverse engineering.

Contributions The main contributions of this work can be summarized as follows:

1. We introduce a real-world CAD reverse engineering-inspired approach that can effectively reconstruct fully parametric CAD models from 3D scans while preserving fine-grained geometric details.
2. To the best of our knowledge, this is the first approach capable of reconstructing CAD models from 3D scans while incorporating sketch constraints.
3. We conduct comprehensive experiments on publicly available benchmarks and demonstrate that our method outperforms existing state-of-the-art techniques in CAD reverse engineering.

2 Related Work

3D CAD Reverse Engineering. Early approaches to CAD reconstruction rely on Constructive Solid Geometry (CSG) representations [22, 23, 24, 25, 26, 27, 28], using Boolean operations over primitives, but these are limited in capturing complex, real-world CAD structures. Other works target Boundary Representation (BRep) reconstruction [29, 30, 18, 31], focusing on high-precision surface and topology modeling. More recent efforts adopt sketch-extrude paradigms, which better reflect the way CAD models are constructed in practice. Within this direction, geometry-grounded bottom-up methods [20, 7, 8] estimate sketches and extrusion parameters from input scans, leveraging structural cues from the geometry but often lacking full parametric editability and seamless integration in CAD software [17]. Alternatively, language-based top-down methods [17, 19, 9] learn to generate construction sequences, capturing design intent but struggling to preserve fine-grained geometric fidelity. We propose a practical reverse engineering-inspired approach that reconstructs detailed and editable CAD models with explicit sketch primitives and constraints, directly compatible with standard CAD software.

Constrained Sketches in CAD. In CAD modeling, sketch primitives and their associated constraints form the foundation of design intent [32, 33], as they govern how a design adapts when modified. Parametric CAD sketch primitives (e.g., lines, arcs, circles) provide a flexible 2D interface for defining shapes, while constraints (e.g., perpendicular, parallel, tangent) preserve geometric relationships, allowing controlled variation without compromising the overall structure. When designers apply constraints (e.g., orthogonality) to sketch primitives (e.g., two lines), any changes made to one primitive are automatically adjusted to preserve these relationships. For example, if the parameters of one line are modified, the other will update accordingly to maintain orthogonality. In a reverse engineering context, it is highly desirable not only to infer accurate CAD geometry from input scans but also to recover the underlying design intent, ensuring that the reconstructed model responds to modifications in a manner consistent with how a designer would have originally constrained it. The availability of large-scale CAD sketch datasets [34] has facilitated research in this area, with recent work focusing on generative CAD sketch modeling [21, 4, 35], sketch image parameterization [36, 37, 38], and design intent inference [39, 40]. While these methods have achieved promising results for 2D sketches, their effectiveness in 3D CAD modeling remains underexplored.

3 Preliminaries and Problem Statement

In this section, we provide the problem statement along with the preliminary definitions.

Definition 1 (Sketch Primitive)**.** *A sketch primitive* $\mathbf{p}$ *is a 2D entity uniquely defined by geometric parameters. We define the following 2D sketch primitives:* ***Line:*** *Defined by its start point* $(x_s, y_s) \in \mathbb{R}^2$ *and end point* $(x_e, y_e) \in \mathbb{R}^2$*.* ***Circle:*** *Represented by its center* $(x_c, y_c) \in \mathbb{R}^2$ *and radius* $r \in \mathbb{R}$*.* ***Arc:*** *A segment of a circle, specified by its start point* $(x_s, y_s) \in \mathbb{R}^2$*, mid-point* $(x_m, y_m) \in \mathbb{R}^2$*, and end point* $(x_e, y_e) \in \mathbb{R}^2$*, all lying on the same circle.*

Definition 2 (Sketch Constraint)**.** *A sketch constraint* $\mathbf{c} = (\mathbf{p}_i, \mathbf{p}_j, c_t)$ *is defined by a constraint type* c_t *that specifies the relationship between two primitives* $\mathbf{p}_i$ *and* $\mathbf{p}_j$*. The types of constraints considered in this work are: coincident, concentric, equal, fix, horizontal, midpoint, normal, offset, parallel, perpendicular, quadrant, tangent, and vertical. Note that some constraints are defined on a single primitive such as vertical and horizontal.*

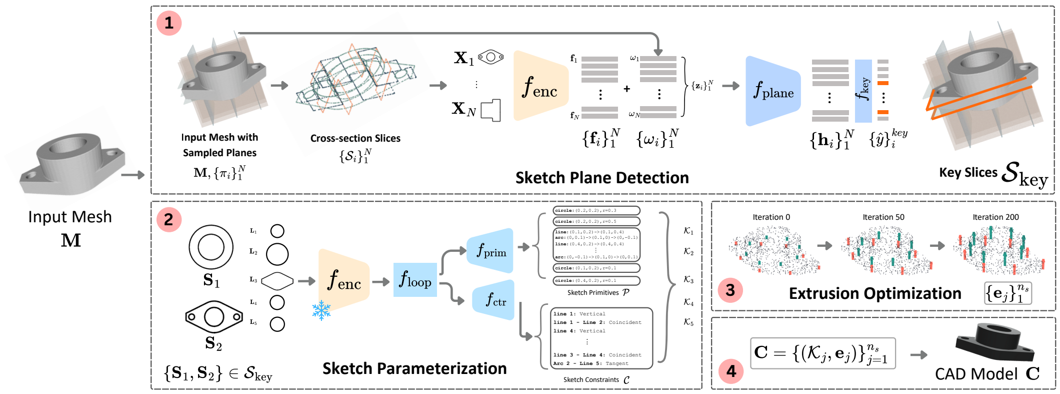

Figure 2: Overview of the method. `MiCADangelo` comprises three main components: Sketch Plane Detection, Sketch Parameterization, and Differentiable Extrusion. The generated constrained sketches, together with the optimized extrusion parameters, are assembled into the final parametric CAD model.

Definition 3 (Constrained Sketch)**.** *A constrained sketch* $\mathcal{K} = (\mathcal{P}, \mathcal{C})$ *is defined by a set of* n_p *2D sketch primitives* $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{n_p}$ *and a set of* n_c *sketch constraints* $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^{n_c}$.

Definition 4 (Sketch Plane)**.** *A sketch plane* π *is defined by an origin point* $\mathbf{o} \in \mathbb{R}^3$ *and a normal vector* $\mathbf{n} \in \mathbb{R}^3$*, on which a sketch is defined and centered at* $\mathbf{o}$.

Definition 5 (3D Mesh)**.** *A 3D mesh is defined as* $\mathbf{M} = (\mathcal{V}, \mathcal{F})$ *where* $\mathcal{V} \subset \mathbb{R}^3$ *are vertices and* $\mathcal{F} \subset \mathcal{V}^3$ *are triangles formed by 3 vertices.*

Definition 6 (Cross-section Slice)**.** *Given a 3D mesh* $\mathbf{M}$ *and a slicing plane* π*, a cross-section slice* $\mathcal{S}$ *is formed by sets of line segments resulting from the intersection of the mesh triangles with* π*. This slicing process naturally yields multiple connected components, each corresponding to a distinct set of connected line segments. Formally, the cross-section slice* $\mathcal{S}$ *can be represented as a collection of* L *line sets* $\{\mathbb{L}_j\}_{j=1}^{L}$*, where each line set composed of* n_l *successive points* $\{\mathbf{q}_i\}_{i=1}^{n_l} \in \mathbb{R}^3$ *is defined as* $\mathbb{L}_j = \{(\mathbf{q}_i, \mathbf{q}_{i+1}) \mid i = 1, \ldots, n_l - 1\}$.

Definition 7 (Closed Loop)**.** *A closed loop* $\mathbf{L}$ *is a set of line segments connecting* n_l *successive points* $\{\mathbf{q}_i\}_{i=1}^{n_l} \in \mathbb{R}^3$ *forming a non-self-intersecting and enclosed region. It is defined as:* $\mathbf{L} = \{(\mathbf{q}_i, \mathbf{q}_{i+1}) \mid i = 1, \ldots, n_l - 1\} \cup \{(\mathbf{q}_{n_l}, \mathbf{q}_1)\}$*. It is important to note that, in the context of CAD models, the line sets forming the cross-section slice* $\mathcal{S}$ *defined in the above definition are usually closed loops.*

Definition 8 (Extrusion)**.** *An extrusion* $\mathbf{e} = (\pi, t, \mathbf{v}, h)$ *is defined by a sketch plane* π*, an extrusion type* t*, and extrusion parameters* $(\mathbf{v}, h)$*, where* $\mathbf{v} \in \mathbb{R}^3$ *is the extrusion direction vector and* $h \in \mathbb{R}$ *is the extrusion length. The extrusion operation extends a sketch* $\mathcal{K}$ *along* $\mathbf{v}$ *for a distance* h*, forming a 3D solid. The extrusion type* t *specifies whether the operation creates material (new) or removes material (cut).*

Definition 9 (CAD Model Representation)**.** *A CAD model* $\mathbf{C}$ *is defined as the sequential combination of* n_s *sketch-extrude steps* $\{(\mathcal{K}_i, \mathbf{e}_i)\}_{i=1}^{n_s}$.

Problem Statement. Given an input 3D mesh $\mathbf{M}$, the goal is to recover the sequence of n_s sketch-extrude steps reconstructing the CAD model $\mathbf{C} = \{(\mathcal{K}_j, \mathbf{e}_j)\}_{j=1}^{n_s}$. Each extrusion $\mathbf{e}_j = (\pi_j, t_j, \mathbf{v}_j, h_j)$ is defined by a sketch plane π_j, an extrusion type t_j, and extrusion parameters $(\mathbf{v}_j, h_j)$. Each constrained sketch $\mathcal{K}_j = (\mathcal{P}_j, \mathcal{C}_j)$ consists of 2D parametric primitives $\mathcal{P}_j$ along with their associated CAD constraints $\mathcal{C}_j$.

4 Proposed Method

4.1 Method Overview

As illustrated in Figure 1, `MiCADangelo` is inspired by the way human designers approach CAD reverse engineering. Given an input mesh $\mathbf{M}$ obtained from a 3D scan, the key idea—mirroring human intuition—is to identify the most relevant cross-section slices that enable accurate reconstruction of the original CAD model. To achieve this, `MiCADangelo` samples N equally-spaced slicing planes $\{\pi_i\}_{i=1}^{N}$ along the x-, y-, and z-axes, producing a set of 2D cross-section slices $\{\mathcal{S}_i\}_{i=1}^{N}$. These slices are then converted into raster images and supplied with contextual embeddings before being passed into a sketch plane detection network that identifies the most relevant slices $\mathcal{S}_{\text{key}} = \{\mathcal{S}_i\}_{i=1}^{n_{key}}$ (Section 4.2). Once the key cross-section slices are identified—again following the logic used by human designers—the next step is to predict the sketch primitives and their associated constraints from these slices to obtain the corresponding constrained sketches. To achieve this, each key cross-section slice is decomposed into separate closed loops $\{\mathbf{L}_j\}_{j=1}^{L}$. Each loop is converted into a raster image and passed to a network for constrained sketch parameterization with the goal of inferring the corresponding sketch primitives and constraints $\{\mathcal{K}_j\}_{j=1}^{L}$ (Section 4.3). After obtaining the constrained sketches, a differentiable extrusion optimization is performed for each sketch $\mathcal{K}_j$ *w.r.t* the input geometry of the mesh $\mathbf{M}$ to find out the corresponding extrusion parameters $\mathbf{e}_j$ (Section 4.4). Finally, the obtained extruded elements are assembled together to obtain the final CAD model. An overview of the proposed method is illustrated in Figure 2.

4.2 Sketch Plane Detection Network

Given a set of cross-section slices $\{\mathcal{S}_i\}_{i=1}^{N}$ extracted from the input mesh $\mathbf{M}$, our goal is to identify a subset of these as key sketch plane slices. Each slice $\mathcal{S}_i$ is projected onto a 2D plane and normalized to fit within a unit bounding box, which is subsequently rendered as a binary image $\mathbf{X}_i \in \{0,1\}^{H\times W}$. The slice image $\mathbf{X}_i$ is passed through a convolutional ResNet34 encoder $f_{\text{enc}} : \{0,1\}^{H\times W} \rightarrow \mathbb{R}^d$ to produce a d-dimensional latent vector $\mathbf{f}_i = f_{\text{enc}}(\mathbf{X}_i)$. In order to contextualize the embedding of the different slices and maintain their spatial relationships, each slice is associated with a slice index $\sigma_i \in \{0,\ldots,N-1\}$, axis identifier $a_i \in \{0,1,2\}$ for x-, y-, or z-axis, normalization parameters $\boldsymbol{\eta}_i = (t_i^x, t_i^y, s_i) \in \mathbb{R}^3$ that correspond to the translation and the scale parameters of the normalization process. These inputs are embedded into the latent space $\mathbb{R}^d$,

$$\omega_i^{\text{pos}} = \mathbf{W}_{\text{pos}}\mathbf{y}_{\sigma_i}, \quad \omega_i^{\text{axis}} = \mathbf{W}_{\text{axis}}\mathbf{y}_{a_i}, \quad \omega_i^{\text{norm}} = \mathbf{W}_{\text{norm}}\boldsymbol{\eta}_i \ , \tag{1}$$

where $\mathbf{y}_{\sigma_i} \in \{0,1\}^N$ and $\mathbf{y}_{a_i} \in \{0,1\}^3$ are one-hot vectors for slice index and axis identifier, $\mathbf{W}_{\text{pos}} \in \mathbb{R}^{d\times N}$ and $\mathbf{W}_{\text{axis}} \in \mathbb{R}^{d\times 3}$ are the corresponding learnable matrices, and $\mathbf{W}_{\text{norm}} \in \mathbb{R}^{d\times 3}$ is a linear projection of normalization parameters. The final embedding $\mathbf{z}_i \in \mathbb{R}^d$ is obtained as

$$\mathbf{z}_i = \mathbf{f}_i + \omega_i \ , \tag{2}$$

where the contextual embedding is defined as $\omega_i = \omega_i^{\text{pos}} + \omega_i^{\text{axis}} + \omega_i^{\text{norm}}$. Next, the set of embedding $\{\mathbf{z}_i\}_{i=1}^{N}$ is fed into a multi-layer multi-head transformer encoder $f_{\text{plane}} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ to obtain a set of richer embedding $\{\mathbf{h}_i\}_{i=1}^{N} \in \mathbb{R}^d$ as follows,

$$(\mathbf{h}_1,\ldots,\mathbf{h}_N) = f_{\text{plane}}(\mathbf{z}_1,\ldots,\mathbf{z}_N) \ . \tag{3}$$

Each encoded slice embedding $\mathbf{h}_i$ is passed through a binary classifier $f_{\text{key}} : \mathbb{R}^d \rightarrow [0,1]$ composed of a linear layer and a sigmoid to predict the probability of being a key sketch plane as follows,

$$\hat{y}_i^{\text{key}} = f_{\text{key}}(\mathbf{h}_i) \ . \tag{4}$$

Note that the sketch plane detection network is trained by a binary cross-entropy loss using supervised key sketch plane labels from the DeepCAD dataset [6]. The process of obtaining these labels is explained in the supplementary. Finally, the subset $\mathcal{S}_{\text{key}} = \{\mathcal{S}_i \mid \hat{y}_i^{\text{key}} \geq \tau\}$ is determined to be the set of key slices with the corresponding planes π_i, where τ is a fixed threshold set to 0.5.

4.3 Constrained Sketch Parameterization Network

Once the key cross-section slices $\mathcal{S}_{\text{key}}$ are obtained by the sketch plane detection network, the goal of the constrained sketch parameterization network is to infer the constrained parametric sketches $\{\mathcal{K}_j\}_{j=1}^{L}$ from the individual closed loops $\{\mathbf{L}_j\}_{j=1}^{L}$ of each key cross-section slice. For each $\mathcal{S}_i \in \mathcal{S}_{\text{key}}$, the set of resulting closed loops $\{\mathbf{L}_j\}_{j=1}^{L}$ are projected to 2D and rendered as binary images $\{\mathbf{X}_j\}_{j=1}^{L}$. The constrained sketch parameterization network operates directly on these rasterized loop images to infer the corresponding sketch primitives and geometric constraints. This approach enables to preserve fine-grained geometric details of the input mesh, which are essential for accurate CAD reconstruction.

The design of the constrained sketch parameterization network is similar to the one introduced in [37]. Each image $\mathbf{X}_j \in \{0,1\}^{H\times W}$ is then encoded by a convolutional encoder $f_{\text{enc}} : \{0,1\}^{H\times W} \to \mathbb{R}^d$ (same as the one used to encode the cross-section slices in Section 4.2), producing a latent feature vector $\mathbf{f}_j \in \mathbb{R}^d$. The latent vector is then passed to a transformer encoder-decoder $f_{\text{loop}} : \mathbb{R}^d \to \mathbb{R}^{d_e\times n_p}$ to obtain a set of embedding for n_p sketch primitives $\zeta_j \in \mathbb{R}^{d_e\times n_p}$ as follows,

$$\zeta_j = f_{\text{loop}}(\mathbf{f}_j) \; . \tag{5}$$

As in [37], the set of embedding is passed to two separate heads: a parameterization head $f_{\text{prim}} : \mathbb{R}^{d_e\times n_p} \to \mathbb{Q}^{n_p}$ and a constraint prediction head $f_{\text{ctr}} : \mathbb{R}^{d_e\times n_p} \to \mathbb{Q}^{n_c}$, where $\mathbb{Q}^{n_p}$ and $\mathbb{Q}^{n_c}$ denote the quantized space of n_p primitive values and n_c constraint values, respectively. These heads result in the parametric sketch primitives and their constraints as follows,

$$\mathcal{P}_j = f_{\text{prim}}(\zeta_j) \quad , \quad \mathcal{C}_j = f_{\text{ctr}}(\zeta_j) \; . \tag{6}$$

For further details on the quantization spaces of primitives and constraints, the design of the constrained sketch parameterization network, and the loss functions employed during training, readers are referred to [37] as well as the supplementary materials for additional explanations. Due to the lack of CAD constraint annotations in the DeepCAD dataset [6] and other 3D CAD modeling datasets, the constrained sketch parameterization network is initially trained on the SketchGraphs dataset [34]. It is then fine-tuned on an augmented version of SketchGraphs that includes added noise and synthetically generated closed-loop images, enabling better adaptation to the characteristics of cross-sectional slices. Note that both the sketch parameterization network and the sketch plane detection network share the same encoder f_{enc}. The encoder is initially trained as part of the sketch parameterization network and then kept frozen during parameterization, while it is fine-tuned during the training of the sketch plane detection network.

4.4 Differentiable Extrusion Optimization

For each constrained sketch $\mathcal{K}_j$ corresponding to a loop $\mathbf{L}_j$ within a key cross-section slice $\mathcal{S}_i$, the goal of differentiable extrusion optimization is to recover the parameters of the extrusion $\mathbf{e}_j$ that best fits the resulting extruded solid to the 3D input mesh. As mentioned in Definition 8 of Section 3, the extrusion is defined by $\mathbf{e}_j = (\pi_j, t_j, \mathbf{v}_j, h_j)$, with π_j denotes the sketch plane from which the extrusion originates, $t_j \in \{\textit{new}, \textit{cut}\}$ specifies the extrusion type, and $(\mathbf{v}_j, h_j)$ represent the extrusion direction and length, respectively.

The plane π_j is determined by the key cross-section slice associated with the input sketch $\mathcal{K}_j$ and its normal vector is used to define the direction of the extrusion $\mathbf{v}_j$. The extrusion type t_j is assigned based on the nesting hierarchy of the loop $\mathbf{L}_j$ within its corresponding cross-section slice $\mathcal{S}_i$. The outermost loops are labeled as *new* extrusions, and the label alternates with each subsequent level of nesting: a loop contained within a *new* extrusion loop is labeled as *cut* extrusion, a loop within a *cut* extrusion loop is labeled as *new* extrusion, and this alternating pattern continues with increasing depth. For loops labeled as *new*, the corresponding length h_j is determined through an extrusion optimization process. Loops labeled as *cut* are interpreted as infinite cuts.

Extrusion Length Optimization. To optimize over the extrusion length h_j, we define a set of *extrusion vectors* over the sketch $\mathcal{K}_j$ by sampling a set of n_r 3D anchor points $\{\mathbf{r}_k\}_{k=1}^{n_r} \in \mathbb{R}^3$ along the loop boundary. Each anchor point is associated with a learnable scalar extrusion length $h_j \in \mathbb{R}$, shared across the anchor points of the same loop. Each anchor point $\mathbf{r}_k$ is mapped to an extrusion vector $\rho_k \in \mathbb{R}^3$ along the direction of the extrusion $\mathbf{v}_j$ with a length h_j using the following function,

$$\mathbf{F} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{F}(\mathbf{r}_k, h_j\mathbf{v}_j) := \mathbf{r}_k + h_j\mathbf{v}_j = \rho_k \; . \tag{7}$$

This results in a set of extrusion vectors $\{\rho_k\}_{k=1}^{n_r}$, each of them has its corresponding anchor point $\mathbf{r}_k$ as starting point and all of them sharing the same length and direction h_j and $\mathbf{v}_j$. Next, a set of n_M points $\mathcal{Q} = \{\mathbf{q}_l\}_{l=1}^{n_M} \in \mathbb{R}^3$ are sampled on the input mesh $\mathbf{M}$ and the set of extrusion vectors $\{\rho_u\}_{u=1}^{n_r+n_L+n_{key}}$ from each loop and each key cross-section slice are considered. The *point-to-vector*

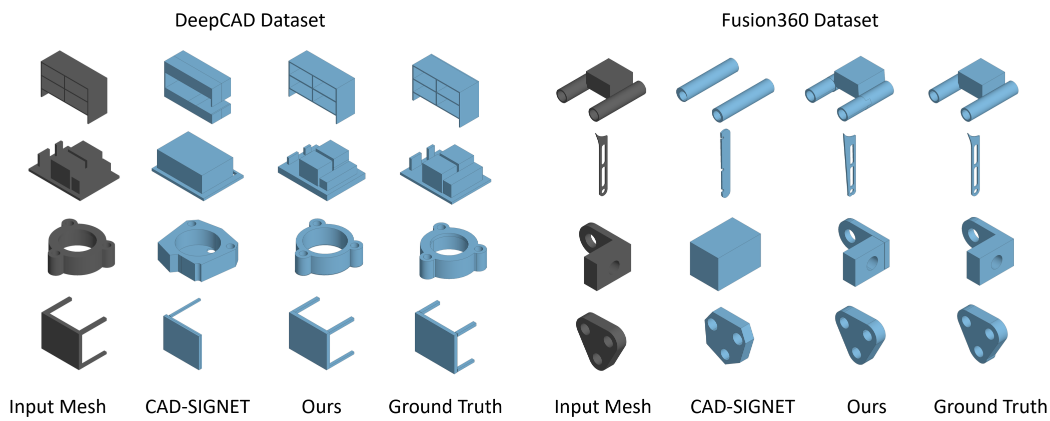

Figure 3: Qualitative comparison of of our method and that of [17] on DeepCAD and Fusion360.

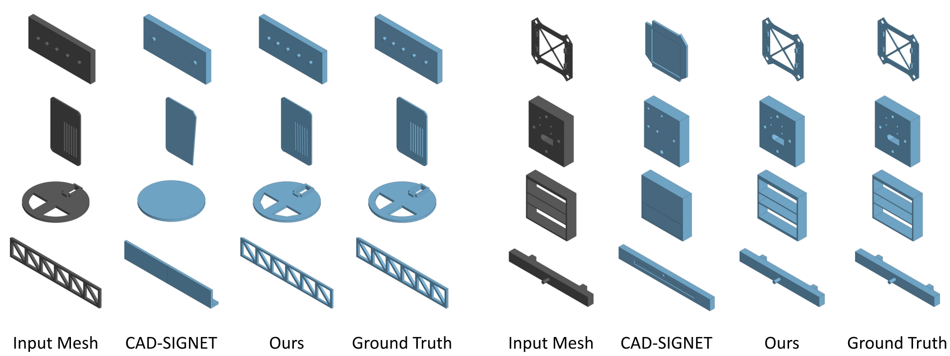

Figure 4: Qualitative comparison between our method and that of [17] on complex models containing fine-grained geometric details.

distance $d(\mathbf{q}_l, \rho_u)$ is then computed between each point $\mathbf{q}_l$ and extrusion vector ρ_u to identify the closest extrusion vector $\rho_{\text{min}} := \arg\min_u \; d(\mathbf{q}_l, \rho_u)$.

The total loss is defined as the mean squared distance over all sampled points, regularized by the squared sum of extrusion lengths to avoid trivial solution. It is given by,

$$\mathcal{L}_{\text{extr}} = \frac{1}{n_M} \sum_{l=1}^{n_M} d(\mathbf{q}_l, \rho_{\text{min}})^2 + \lambda \sum_{i,j} h_j^2 \; , \tag{8}$$

where λ denotes a scaling factor for the regularization term, i refers to the detected key slice plane and j refers to the individual sketch loop. The extrusion lengths h_j are optimized jointly for all the sketch loops over the detected key slice planes via gradient descent to minimize $\mathcal{L}_{\text{extr}}$, thereby adjusting the extrusion vectors ρ_u to best fit to the input mesh geometry.

5 Experiments

In this section, we detail the experimental setup and report results to evaluate the effectiveness of the proposed `MiCADangelo`.

Datasets. We evaluate our method on the test sets of DeepCAD [6] and Fusion360 [18] datasets. The sketch plane detection is trained on the train set of DeepCAD [6], while the constrained sketch parameterization network is trained on the train set of the SketchGraphs [35] dataset and finetuned on an augmented version of the dataset. Details on the processing of the datasets are in the supplementary.

Metrics. For 3D CAD reconstruction, median Chamfer Distance, Intersection over Union (IoU), median Edge Chamfer Distance (ECD) and Invalidity Ratio (IR) are used. For 2D sketches, image-level sketch chamfer distance (SCD) is used. Additional metric details are provided in the supplementary.

Table 1: Quantitative Results on DeepCAD and Fusion360 Datasets.

Method	DeepCAD Test Set				Fusion360 Test Set			
	Med. CD↓	IoU↑	IR↓	ECD↓	Med. CD↓	IoU↑	IR↓	ECD↓
DeepCAD [6]	9.64	46.7	7.1	–	89.2	39.9	25.2	–
Point2Cyl [8]	4.27	73.8	3.9	–	4.18	67.5	3.2	–
CAD-Diffuser [9]	3.02	74.3	1.5	–	3.85	63.2	1.7	–
CAD-SIGNet [17]	0.28	77.6	**0.9**	0.74	0.56	65.6	**1.6**	4.14
Ours	**0.20**	**80.6**	2.6	**0.46**	**0.48**	**68.7**	3.2	**2.66**

Implementation Details. Planes are preprocessed to ensure consistent normal directions along the positive axes. The plane detection model is trained for 20 epochs on DeepCAD [6] with $l_r = 1 \times 10^{-4}$. The constrained sketch parameterization model is trained on SketchGraphs [35] as in [37] and finetuned for 50 epochs on synthetically generated, noise-augmented loops. We use 40 cross-sections per axis, each normalized to an image of size 128×128. Normalization is performed using 2D offsets and a scale factor corresponding to a unit bounding box. The transformer encoder of the sketch plane detection comprises 4 layers and 4 attention heads, with an embedding dimension of 256. The sketch parameterization network architecture is similar to [37]. For the extrusion optimization, we run 200 iterations with a learning rate of 2×10^{-4}. AdamW is used as an optimizer for all the experiments. More details are in the supplementary.

5.1 Comparison with state-of-the-art

The proposed method is compared with DeepCAD [6], CAD-Diffuser [9], Point2Cyl [8], and CAD-SIGNet [17] on the DeepCAD and Fusion360 test sets. A quantitative comparison with these methods is provided in Table 1. Across datasets, `MiCADAngelo` achieves superior reconstruction performance. Note that low IR reported by the best performing [17] are enabled by test-time sampling, where multiple CAD reconstruction candidates are generated and the best is selected. Without test-time sampling, [17] yields an IR of 4.4 and 9.3 for DeepCAD and Fusion360, respectively. Figure 3 provides a qualitative comparison with [17].

Table 2: Quantitative results on complex models and on models with more than two extrusions.

Method	Models with ≥ 4 Loops				Models with > 2 Extrusions			
	Med. CD↓	IoU↑	IR↓	ECD↓	Med. CD↓	IoU↑	IR↓	ECD↓
CAD-SIGNet [17]	1.34	49.2	**3.2**	4.75	3.95	40.6	5.4	9.81
Ours	**0.37**	**68.3**	4.1	**2.04**	**0.46**	**64.8**	**3.0**	**2.27**

Our method consistently generates CAD models that closely resemble the ground-truth geometry across a diverse range of samples. We also evaluate the performance of our method against [17] on a subset of complex models with fine-grained details (containing 4 or more loops) and models with more than 2 extrusions from DeepCAD. Results in Table 2 demonstrate that our methods significantly outperforms [17] in preserving fine-grained geometric details and handling complex geometries. More qualitative results on complex models are provided in Figure 4.

5.2 Impact of Constraints in 3D Reverse Engineering

Table 3: Quantitative results of deformation robustness under sketch modifications

Method	Med. CD↓	IoU↑	IR↓	ECD↓
CAD-SIGNet [17]	2.89	57.4	**3.5**	20.43
Ours	**0.38**	**81.1**	4.3	**1.29**

`MiCADAngelo` is the first CAD reverse engineering method to generate parametric constraints. The proposed explicit modeling of the reverse engineering pipeline enables sketch constraint inference directly from rasterized sketch images, significantly reducing the solution space compared to predicting constraints from point clouds. To demonstrate the importance of constraint modeling in 3D CAD reconstruction, we conduct an experiment analyzing how CAD reconstructions produced by `MiCADAngelo` and [17] respond to sketch modifications similar to those typically performed by CAD designers. Specifically, we introduce a small random displacement to a point in the recovered sketch geometry and evaluate how this change affects the resulting solid

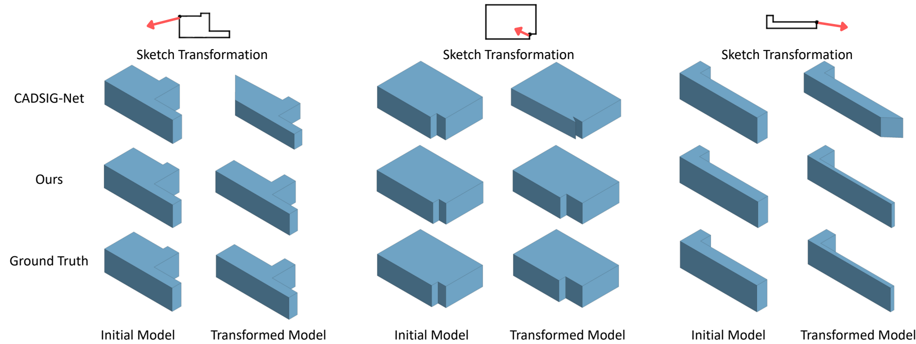

Figure 5: Qualitative comparison of the impact of the constraints by transformations applied on sketch points.

Table 4: Effect of contextual embeddings on plane detection performance.

Contextual Emb.	Precision	Recall	F1
✗	0.317	0.292	0.296
✓	**0.894**	**0.864**	**0.870**

Table 5: Plane detection performance across datasets.

Dataset	Precision	Recall	F1
DeepCAD	0.894	0.864	0.870
Fusion360	0.860	0.812	0.820
CC3D	0.803	0.790	0.777

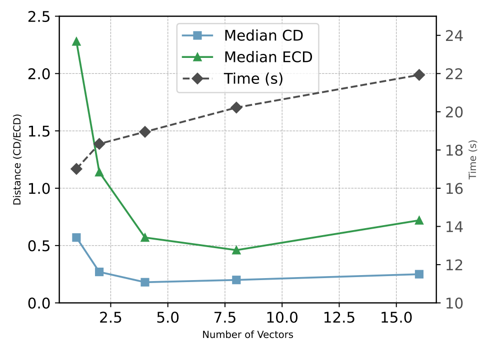

Figure 6: CD and ECD performance and inference times as a function of the number of extrusion vectors.

geometry, comparing it to a constrained ground-truth model modified with the same displacement. Due to unavailability of CAD constraints in DeepCAD and Fusion360 datasets, we form data for this experiment by extruding $1k$ sketches with closed loops from the SketchGraphs dataset [34]. The impact of the displacement on the solid geometry is computed by integration with the FreeCAD API [3].

We compare our method to that of [17]. Note that the output representation of [17] generates closed sketch loops by construction, which implicitly enforces coincident constraints between sketch vertices. Results are reported in Table 3. `MiCADAngelo` reverse-engineered models closely match the ground truth, even after random sketch modifications. As illustrated in Figure 5, due to effective application of CAD sketch constraints, edits in our method propagate correctly through sketch primitives, resulting in modified CAD models that retain structural consistency *w.r.t* the ground-truth. In contrast, coincident constraints alone are insufficient to preserve overall geometry for [17] with displacements leading to geometric distortions. Note that learning sketch constraints might also impact sketch parameterization and the overall CAD reconstruction (e.g., inferring orthogonality constraint can help in enforcing the parameters of the predicted lines to be orthogonal). While constraints are not strictly necessary for sketch primitive parameterization, prior work [35] has shown that jointly learning primitives and constraints benefits both tasks through shared structural information. We leave further exploration of improved geometry with constraints to future work.

5.3 Additional Experiments

We include additional experiments conducted to further evaluate `MiCADangelo`.

Plane Detection. Table 4 ablates the effect of contextual embeddings in slice encoding. Incorporating positional information to the geometric features of the cross-section leads to improved performance. Table 5 shows robust plane detection performance with minimal drop in Fusion360 and CC3D datasets.

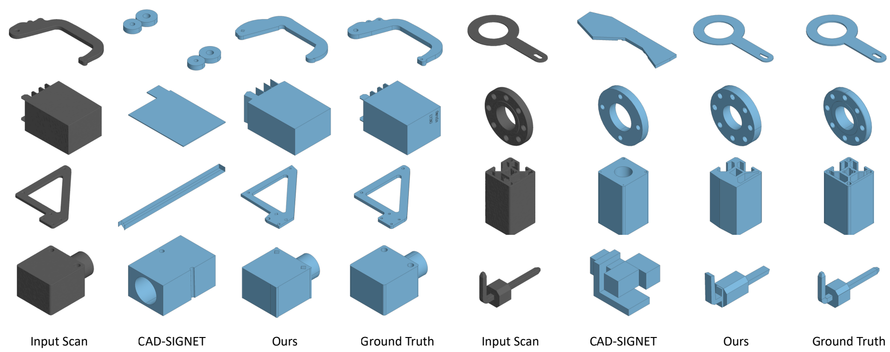

Figure 7: Qualitative comparison of the performance on the real-world scans from CC3D.

Constrained Sketch Parameterization. We evaluate our constrained sketch parameterization network against the recent proposed [37]. To ensure that ground-truth sketches correspond to valid cross-sections, this experiment is conducted on single-extrusion models from the DeepCAD test set. Due to fine-tuning on synthetically generated noisy closed loops, `MiCADangelo` outperforms [37].

Table 6: Evaluation of sketch parameterization on single-extrusion CAD models.

Method	*Avg. SCD*
Davinci [37]	0.827
Ours	**0.283**

Extrusion Optimization. Figure 6 reports differentiable extrusion performance for a varying number of vectors. Performance improves steadily up to 8 vectors, with minimal impact on inference time.

Robustness to Real-world Scans. To evaluate performance on real imperfect scans, we include a quantitative comparison on the challenging CC3D [41, 42] dataset. This cross-dataset evaluation compares our method to that of [17], both trained on DeepCAD and evaluated on CC3D scans that include realistic scanning artifacts, such as holes and misoriented normals. Results are shown in the Table 7. The proposed MiCADangelo outperforms [17] in this in-the-wild setting, demonstrating greater robustness to real-world noise. Figure 7 presents a qualitative evaluation on real-world 3D scans from the CC3D [41, 42] dataset. The results demonstrate the robustness of our approach in handling real-world artifacts and producing cleaner, more accurate CAD models compared to [17].

Table 7: Comparison between CADSIG-Net and our method on CC3D dataset.

Method	Med. CD↓	IoU↑	IR↓	ECD↓
CADSIG-Net [17]	2.90	42.6	4.4	8.68
Ours	**1.69**	**50.8**	**2.2**	**5.93**

6 Conclusion

In this work we proposed `MiCADAngelo`, a novel CAD reverse engineering approach that transforms 3D scans into fully parametric CAD models. Our method is inspired by real-world CAD practices and uniquely incorporates cross-sections and sketch primitives with constraints, enabling the preservation of both high-level parametric structure and fine-grained geometric detail. Evaluation is performed on standard benchmarks were our approach outperforms existing methods establishing a new state-of-the-art in CAD model reconstruction from 3D scans.

Limitations. A detailed discussion on failure cases and limitations is provided in supplementary. Among CAD operations, the proposed method only supports extrusion as in previous works [17, 20, 8]. Extrusions are currently defined based on sketch plane normals, which can be suboptimal for models with non-axis-aligned extrusions. Note that such complex models are also challenging for [6, 9, 17]. The current method does not yet support complex sketch primitives such as B-splines. Future works will address these limitations.

7 Acknowledgements

This work is supported by the National Research Fund (FNR), Luxembourg, under the BRIDGES2021/IS/16849599/FREE-3D project and by Artec3D.

References

[1] Dassault Systèmes. Solidworks, 2025. URL `https://www.solidworks.com`.

[2] PTC Inc. Onshape, 2025. URL `https://www.onshape.com`.

[3] FreeCAD Community. Freecad, 2025. URL `https://www.freecad.org`.

[4] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. In *ICLR*, 2022.

[5] Polyga Inc. Reverse engineering 101: Scan to cad, 2025. URL `https://www.polyga.com/reverse-engineering-101-scan-to-cad/`. Accessed: 2025-05-13.

[6] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *ICCV*, 2021.

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

[8] Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *CVPR*, 2022.

[9] Weijian Ma, Shuaiqi Chen, Yunzhong Lou, Xueyang Li, and Xiangdong Zhou. Draw step by step: Reconstructing cad construction sequences from point clouds via multimodal diffusion. In *CVPR*, 2024.

[10] Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. Reconstructing editable prismatic cad from rounded voxel models. In *SIGGRAPH Asia*, 2022.

[11] Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds. In *ICCV*, 2025.

[12] Dimitrios Mallis, Ahmet Serdar Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-assistant: tool-augmented vllms as generic cad task solvers. In *ICCV*, 2025.

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

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

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

[16] 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 *CVPRW*, 2023.

[17] Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *CVPR*, 2024.

[18] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.

[19] Xiang Xu, Pradeep Kumar Jayaraman, Joseph George Lambourne, Karl D. D. Willis, and Yasutaka Furukawa. Hierarchical neural coding for controllable CAD model generation. In *ICML*, 2023.

[20] Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *CVPR*, 2023.

[21] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. In *NeurIPS*, 2021.

[22] 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)*, 37(6), 2018.

[23] 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*, 2019.

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

[25] Constructive Solid Geometry. Neural shape parsers for constructive solid geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5), 2022.

[26] 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 *CVPR*, 2022.

[27] Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. D^2 csg: Unsupervised learning of compact csg trees with dual complements and dropouts. In *NeurIPS*, 2023.

[28] 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.

[29] 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 *CVPR*, 2021.

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

[31] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *CVPR*, 2021.

[32] Yingzhong Zhang and Xiaofang Luo. Design intent information exchange of feature-based cad models. *WRI CSIE*, 2009.

[33] Jeffrey M. Otey, Manuel Contero, and Jorge D. Camba. Revisiting the design intent concept in the context of mechanical cad education. *Computer-aided Design and Applications*, 15(1), 2018.

[34] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *ICMLW*, 2020.

[35] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. In *NeurIPS*, 2021.

[36] Ahmet Serdar Karadeniz, Dimitrios Mallis, Nesryne Mejri, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Picasso: A feed-forward framework for parametric inference of cad sketches via rendering self-supervision. In *WACV*, 2025.

[37] Ahmet Serdar Karadeniz, Dimitrios Mallis, Nesryne Mejri, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Davinci: A single-stage architecture for constrained cad sketch inference. In *BMVC*, 2024.

[38] Sifan Wu, Amir Hosein Khasahmadi, Mor Katz, Pradeep Kumar Jayaraman, Yewen Pu, Karl Willis, and Bang Liu. Cadvlm: Bridging language and vision in the generation of parametric cad sketches. In *ECCV*, 2024.

[39] Yuezhi Yang and Hao Pan. Discovering design concepts for cad sketches. In *NeurIPS*, 2022.

[40] Evan Casey, Tianyu Zhang, Shu Ishida, John Roger Thompson, Amir Khasahmadi, Joseph George Lambourne, Pradeep Kumar Jayaraman, and Karl DD Willis. Aligning constraint generation with design intent in parametric cad. *arXiv preprint arXiv:2504.13178*, 2025.

[41] Kseniya Cherenkova, Djamila Aouada, and Gleb Gusev. Pvdeconv: Point-voxel deconvolution for autoencoding cad construction in 3d. In *ICIP*, 2020.

[42] 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 *CVPRW*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims presented in the abstract and introduction accurately represent the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 includes a discussion of the limitations, with a more detailed analysis provided in the supplementary material.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include any theoretical proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The detailed setting of the experiments are provided in Section 5 and the supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The data used in this study is publicly available. Scripts for preprocessing the data and extracting additional information used in our work will also be made publicly accessible. However, the code for the method and training cannot currently be released under an open-source license that permits use, modification, and distribution, due to constraints related to the industrial collaboration under which this work was conducted. Nonetheless, all necessary details for reproducing the results are thoroughly provided in the paper and the accompanying supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides sufficient details on the training and testing procedures, evaluation metrics, hyperparameters, and optimizer choice, enabling other researchers to potentially reproduce the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The experiments presented in the paper are computationally intensive. Furthermore, repeated training runs using the same hyperparameters showed no significant variation in validation metrics.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper clearly states the compute resources used for the experiments and the compute time of the method in Section 5 and the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: The paper aims to appropriately highlight the potential positive societal impact in the problem introduction and its motivation. The authors do not anticipate any negative consequences resulting from their work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The authors do not anticipate any potential risks associated with the release of their method, data, or model.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets used in this work, including data and code, are properly cited within the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The extended versions of the public datasets are clearly described in the paper and the supplementary.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work did not involve human subjects or the use of crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work did not involve human subjects or the use of crowdsourcing.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The development of the method does not include any LLMs as a component.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.