CADOps-Net: Jointly Learning CAD Operation Types and Steps from Boundary-Representations

Elona Dupont[1], Kseniya Cherenkova[1, 2], Anis Kacem[1], Sk Aziz Ali[1], Ilya Arzhannikov[2], Gleb Gusev[2], and Djamila Aouada[1]
[1]SnT, University of Luxembourg, Luxembourg
[2]Artec 3D, Luxembourg

Abstract

3D reverse engineering is a long sought-after, yet not completely achieved goal in the Computer-Aided Design (CAD) industry. The ultimate objective is to recover the construction history of a CAD model. Starting from a Boundary Representation (B-Rep) of a CAD model, this paper proposes a new deep neural network, ***CADOps-Net****, that jointly learns the CAD operation types and the decomposition into different CAD operation steps. This joint learning allows to divide a B-Rep into parts that were created by various types of CAD operations at the same construction step; therefore providing relevant information for further recovery of the design history. Furthermore, we propose the novel* ***CC3D-Ops*** *dataset that includes over* $37k$ *CAD models annotated with CAD operation type labels and step labels. Compared to existing datasets, the complexity and variety of CC3D-Ops models are closer to those used for industrial purposes. Our experiments, conducted on the proposed CC3D-Ops and the publicly available Fusion360 datasets, demonstrate the competitive performance of CADOps-Net with respect to state-of-the-art, and confirm the importance of the joint learning of CAD operation types and steps.*

1. Introduction

In today's digital era, Computer-Aided Design (CAD) is the standard option for designing objects ahead of manufacturing [36, 2, 35]. The parametric nature of CAD models allows engineers and designers to iterate over the parameters of existing CAD models to edit and adapt them to new contexts, such as customizing dental prostheses [32], or modifying mechanical parts [34]. However, this is only possible if the final shape of the CAD model comes with its design history. Unfortunately, this is rarely the case as the design history is often not available for generic 3D shapes [6] or lost when CAD models are exchanged between different CAD applications [13, 17]. Consequently, the research community has put a lot of efforts in relating the geometry of 3D shapes to the CAD design history [17, 12, 31, 6, 44, 42]. This process is known as *3D reverse engineering*.

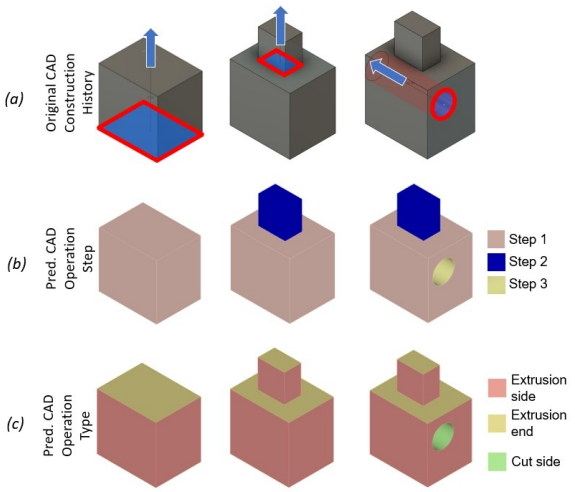


Figure 1: B-Rep segmentation into CAD operations types and steps.

Prior works attempted to recover the CAD design history, considering *Constructive Solid Geometry* (CSG) based models [6, 31] for simplicity. In CSG, a CAD model is represented by a set of rigidly transformed solid primitives (*e.g.* cube, sphere, cylinder) and combined using Boolean operations such as union, intersection, and difference [8]. However, modern CAD workflows use *feature-based* modeling, in which solids are created by iteratively adding features such as holes, slots, or bosses [44, 45]. These high-level features are sequentially created through drawing *sketches* and applying *CAD operations* such as '*extrusion*', '*revolution*', etc. Figure 1a illustrates an example of feature-based

simple CAD model creation. Using this type of CAD modeling, the final model is stored in a data structure called *Boundary-Representation* (B-Rep). The B-Rep describes the geometry and the topology of the CAD model through faces, edges, loops, co-edges and vertices [17]. However, it does not include information about how these entities are designed. Accordingly, recent efforts in the state-of-the-art have focused on relating B-Reps to the design history [44, 17, 12]. In particular, two main directions have been followed: (1) segmenting the B-Rep faces into CAD operation types (*e.g.* '*extrusion*', '*revolution*') [12, 17] or higher-level machining features (*e.g.* '*holes*', '*slots*') [5] that allowed their creation; (2) inferring a sequence of parametric sketches and extrusions that allowed the design of the B-Rep [44, 40, 37]. While the first group of works have the advantage of relating each face of the B-Rep to various types of CAD operations, they do not describe the relationship between the faces nor the steps of the construction. On the other hand, the works taking the second direction reconstruct the ordered sequence of the design history, including sketches, but they are usually limited to only one CAD operation type (*i.e.* '*extrusion*') as a simplification of the search space.

In this work, we combine both directions by segmenting the faces of the B-Reps into various CAD operation types and further decomposing them into steps of construction as shown in Figure 1. These two aspects are jointly learned using an end-to-end neural network, allowing the recovery of further information about the design history such as CAD sketches. The proposed method is evaluated on the publicly available Fusion360 dataset [40], and a newly introduced dataset that is closer to real-world challenges. The key **contributions** can be summarized as follows:

- A neural network, *CADOps-Net*, that operates on B-Reps is proposed to learn the segmentation of faces into CAD operation types and steps. We introduce a joint learning method within an end-to-end model.
- We create a novel dataset, *CC3D-Ops*, that builds on top of the existing CC3D dataset [4] by extending it with B-Reps and their corresponding per-face CAD operation type and step annotations. Compared to existing datasets [40, 23, 14], *CC3D-Ops* better reflects real-world industrial challenges thanks to the complexity of its CAD models. This dataset will be shared with the research community.
- The proposed approach is evaluated on two datasets and compared to recent state-of-the-art methods. We further showcase some preliminary results on a possible downstream application consisting of CAD sketch recovery from B-Reps.

The rest of the paper is organized as follows; We discuss the works related to ours in Section 2 followed by the formulation of problem in Section 3. Section 4 describes the proposed *CADOps-Net*. The proposed *CC3D-Ops* dataset is introduced in Section 5. The experimental results are reported and analyzed in Section 6. Finally, Section 7 concludes this work and presents directions for future work.

2. Related Works

Learning representations for 3D shape modeling is an important research topic that aims at finding the best deep feature encoding method. For instance, while a group of works leverage feature embedding for unordered and irregular point clouds [3, 26, 39, 43, 19] or regular grids of voxels [22, 20, 4, 1, 38], another group of works [10, 9, 21] defines convolution kernels and feature embedding techniques for meshes and manifolds. Other works [6, 17, 12, 31] focused on learning from high-level 3D shape representations such as CAD models. These methods either assume that the CAD models are obtained using CSG or feature-based modeling. In particular, the recovery of the CAD design history considering these two types of modeling has attracted a lot of attention [40, 12, 17, 31, 6].

CSG-based Approaches. Several approaches [31, 27, 8, 41] attempt to infer the design history of CAD models using CSG representation. For instance, when the input shape is a 3D point cloud, [6] and [41] convert it to the CSG tree (mainly binary-tree) of solid bodies which is a volumetric representation of simple geometrical primitives. Similarly, when the input is a B-Rep or a solid body, [30] and [27] describe unique CSG conversion steps (or vice-versa in [8]). The conversion reveals hierarchical steps involved in modeling solid bodies, whereas CAD models appear more as connected surface patches than volumetric solids [28]. Therefore, predicting CSG construction history may not reveal the actual CAD construction steps used in modern CAD workflows [44]. The latter mostly consider B-Reps instead of CSG and rely on feature-based modeling, which is addressed in our work.

Feature-based Approaches. The methods that either directly learn the B-Rep structure of a CAD model [12, 17, 11, 44, 5] or predict sketches and CAD operations [42, 25, 29, 7], are closely related to our work. The works in [25, 7] propose generative models for CAD sketches with a focus on the constraints of sketch entities. Therefore, they do not consider the connection between constrained CAD sketches and operations. On the other hand, methods like SolidGen [11], BRepNet [17], UV-Net [12], CADNet [5] put more emphasis on how to use the B-Rep data structure to obtain face embeddings followed by face segmentation, but obscuring the relation between the segmented faces and design steps. DeepCAD [42], Fusion360 [40] and Zonegraph [44] are the first set of methods, to the best of our knowledge, that relate parametric sketches and CAD operations proposing a generative model for CAD design. How-

ever, their models were restricted to only one type of CAD operations, namely extrusion. Finally, Point2Cyl [37] operates on point clouds to detect 2D sketches but is also limited to the CAD extrusion operation.

CAD Modeling Datasets. Besides Fusion360 [40], there are no datasets that provide both B-Reps and fully explicit construction history in standard format. For example, the ABC dataset [14] provides $1M+$ CAD models with sparse construction history provided in Onshape proprietary format [40]. On the other hand, the SketchGraphs dataset [29] contains a large number of sketch construction sequences but not the B-Reps. Both MFCAD [23] and MFCAD++ [5] datasets contain B-Reps and machining feature labels. However, the samples are synthetic models and too simple to consider for industrial modeling tasks. CC3D dataset [4] offers $50k+$ pairs of industrial CAD models as triangular meshes and their corresponding 3D scans, but without construction steps and B-Reps.

3. Problem Statement

A B-Rep $\mathcal{B}$ can be defined as a tuple of three sets of entities – *i.e.*, a set of N_f faces $\{f_1, f_2, \ldots, f_{N_f}\}$, a set of N_e edges $\{e_1, e_2, \ldots, e_{N_e}\}$, and a set of N_c co-edges (also known as directed half-edges) $\{c_1, c_2, \ldots, c_{N_c}\}$. Our main goal is to relate each face f in $\mathcal{B}$ with its construction history using three different types of features $\mathbf{F} \in \mathbb{R}^{N_f \times d_f}$, $\mathbf{E} \in \mathbb{R}^{N_e \times d_e}$, and $\mathbf{C} \in \mathbb{R}^{N_c \times d_c}$ extracted for the three entities, namely, faces, edges, and co-edges, respectively[1].
The CAD construction history is defined as a sequential combination of sketches followed by some CAD operations. In this work, we are interested in learning (1) the type of CAD operations through the segmentation of each face that allowed for its creation, and (2) the CAD operation step to which the segmented face belongs. Both are detailed below.

3.1. CAD Operation Types

The choice of CAD operation types is crucial for constructing CAD models. For notation simplicity, let us denote them as ***op.types***. The geometry of the final CAD model, usually stored as a B-Rep, is obtained through these operations, which makes each face of the B-Rep directly related to a type of operation. In Figure 1c, we show some intermediate steps of CAD construction and how the faces of the corresponding B-Rep are obtained using different *op.types*. For example, the B-Rep of a cube that was obtained by sketching a 2D square and applying an extrusion operation, as in Figure 1a, would result in two faces with *'extrude end'* labels and four faces with *'extrude side'* labels. The ability to automatically infer the *op.type* that allowed for the creation of each face of the B-Rep constitutes a first, yet essential, step towards relating the geometry of the CAD model to its construction history. Recently introduced models [17, 12] proposed to learn the segmentation of B-Rep faces into *op.types*.
Formally, let us consider a B-Rep $\mathcal{B}$ labelled with the per-face *op.types* $\mathbf{T} = [\mathbf{t}_1; \mathbf{t}_2; \ldots; \mathbf{t}_{N_f}] \in \{0,1\}^{N_f \times k_t}$, where k_t is the number of possible *op.types*. Here, $\mathbf{T} \in \{0,1\}^{N_f \times k_t}$ is an $N_f \times k_t$ matrix with binary entries, where each row $\mathbf{t}_j \in \{0,1\}^{k_t}$ can have only one element as 1 representing the *op.type* of the face f_j. The task of *op.type* segmentation consists of learning a mapping $\mathbf{\Phi}$, such that,

$$\mathbf{\Phi} : \mathbb{R}^{N_f \times d_f} \times \mathbb{R}^{N_e \times d_e} \times \mathbb{R}^{N_c \times d_c} \rightarrow \{0,1\}^{N_f \times k_t} ,$$
$$\mathbf{\Phi}(\mathbf{F}, \mathbf{E}, \mathbf{C}) = \mathbf{T} .$$

It is important to highlight that the segmentation task of *op.types* uses the features of faces, edges and co-edges, but assigns a unique *op.type*, among a fixed number of possible types, to each face of the B-Rep. Despite its usefulness for reconstructing the CAD construction history of B-Reps, the segmentation into *op.types* is not sufficient as it does not describe the relationship between the faces nor the steps of the construction.

3.2. CAD Operation Steps

In addition to the operation types that are assigned to the faces of the B-Reps, our aim is to relate them further to the construction history. Accordingly, we propose a novel task consisting of segmenting the faces of B-Reps into CAD operation steps. For notation simplicity, they will be denoted as ***op.steps*** in what follows. While the segmentation into *op.types* aims at identifying the operation that was used to create each face, the purpose of the segmentation into *op.steps* is to group faces that were created at the same time step. An example of this segmentation is shown in Figure 1b.
Formally, let us consider a B-Rep $\mathcal{B}$ labelled with the per-face *op.steps* $\mathbf{S} \in \{0,1\}^{N_f \times k_s}$, where k_s denotes the number of *op.steps* in $\mathcal{B}$. Similarly to the *op.types* $\mathbf{T}$, the *op.steps* are represented by an $N_f \times k_s$ binary matrix $\mathbf{S} = [\mathbf{s}_1; \mathbf{s}_2; \ldots; \mathbf{s}_{N_f}] \in \{0,1\}^{N_f \times k_s}$. Each row of this matrix, $\mathbf{s}_j \in \{0,1\}^{k_s}$, can have only one element equal to 1 denoting the *op.step* for the face f_j. Segmenting the faces of B-Reps into *op.steps*, would require learning a mapping $\mathbf{\Psi}$,

$$\mathbf{\Psi} : \mathbb{R}^{N_f \times d_f} \times \mathbb{R}^{N_e \times d_e} \times \mathbb{R}^{N_c \times d_c} \rightarrow \{0,1\}^{N_f \times k_s} ,$$
$$\mathbf{\Psi}(\mathbf{F}, \mathbf{E}, \mathbf{C}) = \mathbf{S} .$$

The proposed segmentation into *op.steps* is a challenging task for two main reasons: (1) unlike the *op.type* segmentation where the possible types are predefined, the labels of *op.steps* $\mathbf{S}$ are arbitrary and any combination of labels, in which faces belonging to the same step have identical labels, can be considered as correct; (2) predicting

[1]The considered features are described in Section 6.1.

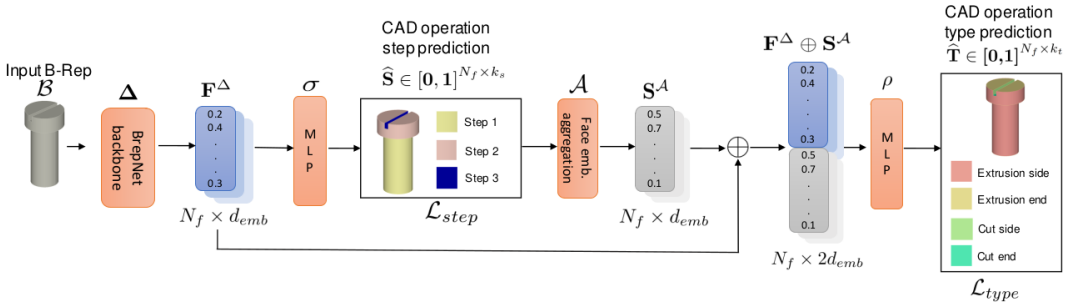


Figure 2: The *CADOps-Net* joint learning network architecture. The input B-Rep, $\mathcal{B}$, is first passed through a BrepNet backbone, $\mathbf{\Delta}$, to obtain face embeddings, $\mathbf{F}^{\Delta}$. These embeddings are then fed to an MLP layer, σ, to predict the face *op.step* segmentation, $\widehat{\mathbf{S}}$. Using these predictions, the face embeddings, $\mathbf{F}^{\Delta}$, are aggregated with a function $\mathcal{A}$ into step embeddings, $\mathbf{S}^{\mathcal{A}}$. Finally the concatenation, $\oplus$, of the face embeddings, $\mathbf{F}^{\Delta}$, and their corresponding step embeddings, $\mathbf{S}^{\mathcal{A}}$, are passed through an MLP layer, ρ to predict the *op.type* face labels.

op.steps aims at grouping B-Rep faces according to the design history. Therefore, it requires learning the relationship between the different faces of the B-Rep in addition to its geometry and topology.

4. Proposed CADOps-Net

The proposed *CADOps-Net* jointly learns the *op.type* and *op.step* segmentation within the same model. In practice, the mappings $\mathbf{\Psi}$ and $\mathbf{\Phi}$, introduced in Section 3, are learnt using an end-to-end neural network. BRepNet [17] is used as the backbone of our model, as it has been shown to effectively operate on B-Reps. BRepNet uses the face, edge, and co-edge features $(\mathbf{F}, \mathbf{E}, \mathbf{C})$ of a B-Rep $\mathcal{B}$ to learn per-face embeddings using a succession of convolutions defined through specific topological walks and Multilayer Perceptron (MLP) layers. For more details about this backbone, readers are referred to [17]. In what follows, the BRepNet backbone will be denoted by $\mathbf{\Delta} : \mathbb{R}^{N_f \times d_f} \times \mathbb{R}^{N_e \times d_e} \times \mathbb{R}^{N_c \times d_c} \rightarrow \mathbb{R}^{N_f \times d_{emb}}$ and $\mathbf{f}^{\Delta}$ will be used as a notation for the embedding extracted using this backbone from a face f of a B-Rep $\mathcal{B}$. The proposed network is composed of two modules that are described below.

4.1. CAD Operation Step Segmentation

The CAD operation step module has two roles. Firstly, it predicts the per-face *op.step* labels. Secondly, it is used to aggregate the embeddings of faces belonging to the same step and produce embeddings for each group of faces obtained in a single *op.step*.

Learning CAD operation steps: The mapping $\mathbf{\Psi}$ introduced in Section 3.2 consists of two components, *i.e.*, $\mathbf{\Psi} := \sigma \circ \mathbf{\Delta}$, where $\mathbf{\Delta}$ uses the features of the B-Rep $(\mathbf{F}, \mathbf{E}, \mathbf{C})$ and extracts per-face embeddings $\mathbf{F}^{\Delta} = [\mathbf{f}_1^{\Delta}; \mathbf{f}_2^{\Delta}; \ldots; \mathbf{f}_{N_f}^{\Delta}] \in \mathbb{R}^{N_f \times d_{emb}}$, and an MLP followed by softmax and maps the face embeddings $\mathbf{F}^{\Delta}$ into probabilities to belong to predicted *op.steps* $\widehat{\mathbf{S}} = [\widehat{\mathbf{s}}_1; \widehat{\mathbf{s}}_2; \ldots; \widehat{\mathbf{s}}_{N_f}] \in [0,1]^{N_f \times k_s}$. Here, each face f_j would have a vector $\widehat{\mathbf{s}}_j \in [0,1]^{k_s}$ specifying its membership probabilities to the k_s *op.steps*. It is important to note that the number of *op.steps* in a CAD model is not known in advance. We assume the maximum number of steps, k_s, in a B-Rep $\mathcal{B}$ to be the largest number of possible steps per model computed on the training dataset.

As mentioned in Section 3.2, a particular challenge for predicting the *op.steps* is that the ground truth labels $\mathbf{S}$ are arbitrary. Therefore, the task consists of predicting the combination of steps that matches the ground truth labels. Inspired by [18, 37], we use a Hungarian matching [16] to find the best one-to-one correspondences between the predicted *op.steps* $\widehat{\mathbf{S}}$ and ground truth labels $\mathbf{S}$. Even though the Hungarian matching is not differentiable, it is only used to find the correspondences in the training phase, allowing for the computation of a *Relaxed Intersection over Union* (RIoU) [15] metric between pairs of predictions $\widehat{\mathbf{s}}$ and ground truth $\mathbf{s}$ as follows,

$$\text{RIoU}(\mathbf{s}, \widehat{\mathbf{s}}) = \frac{\mathbf{s}^{\text{T}}\widehat{\mathbf{s}}}{||\mathbf{s}||_1 + ||\widehat{\mathbf{s}}||_1 - \mathbf{s}^{\text{T}}\widehat{\mathbf{s}}}, \tag{1}$$

where $||.||_1$ denotes the ℓ_1 norm, and $^{\text{T}}$ the vector transpose. The RIoU metric is further used to define the following *op.step* loss function,

$$\mathcal{L}_{step} = \frac{1}{N_f} \sum_{j=1}^{N_f} (1 - \text{RIoU}(\mathbf{s}_j, \widehat{\mathbf{s}}_j)) \,. \tag{2}$$

For inference, the Hungarian matching is not used and the predicted *op.steps* are given by taking the maximum probability over each $\widehat{\mathbf{s}}$.

CAD operation step embedding: In addition to predicting the per-face *op.steps* given a B-Rep, the same module is used to extract CAD step embeddings $\{\mathbf{s}_1^{\mathcal{A}}, \mathbf{s}_2^{\mathcal{A}}, \ldots, \mathbf{s}_{k_s}^{\mathcal{A}}\}$. This is achieved by aggregating the embeddings of faces predicted to belong to the same *op.step*. Specifically, each *op.step* φ would have an embedding $\mathbf{s}_{\varphi}^{\mathcal{A}} \in \mathbb{R}^{d_{emb}}$, such that

$$\mathbf{s}_{\varphi}^{\mathcal{A}} = \underset{j=\arg\max \widehat{\mathbf{S}}_{:,\varphi}}{\mathcal{A}} \mathbf{f}_j^{\Delta} , \tag{3}$$

where $\widehat{\mathbf{S}}_{:,\varphi}$ denotes the per-face predicted *op.step* labels for φ, and $\mathcal{A}$ is an aggregation function that preserves the dimension of the input embeddings such as average or maximum. Finally, each face of the B-Rep will have the corresponding *op.step* embedding $\mathbf{s}^{\mathcal{A}}$ according to the predicted *op.step* label. These embeddings are finally stacked in a matrix $\mathbf{S}^{\mathcal{A}} \in \mathbb{R}^{N_f \times d_{emb}}$.

4.2. CAD Operation Type Segmentation

The introduced mapping $\mathbf{\Phi}$ to obtain the *op.type* segmentation from an input B-Rep shares the same BRepNet backbone $\mathbf{\Delta}$ used by the module of *op.type* segmentation. Moreover, it uses two other mappings, γ and ρ, where $\mathbf{\Phi} := \rho \circ \gamma \circ \mathbf{\Delta}$. The mapping $\gamma : \mathbb{R}^{N_f \times d_{emb}} \times \mathbb{R}^{N_f \times d_{emb}} \rightarrow \mathbb{R}^{N_f \times 2d_{emb}}$ takes as input the face embeddings $\mathbf{F}^{\Delta}$ and outputs their concatenation with the corresponding step embeddings $\mathbf{S}^{\mathcal{A}}$. These concatenated embeddings are fed to an MLP with softmax which are represented by $\rho : \mathbb{R}^{N_f \times 2d_{emb}} \rightarrow \{0,1\}^{N_f \times k_t}$. The final *op.types* $\widehat{\mathbf{T}}$ can be obtained following,

$$\widehat{\mathbf{T}} = \rho(\mathbf{F}^{\Delta} \oplus \mathbf{S}^{\mathcal{A}}) , \tag{4}$$

where $\oplus$ is the column-wise concatenation operation. The loss function for the *op.type* segmentation is computed using the cross-entropy $\mathcal{H}$ between the predicted per-face *op.types* $\widehat{\mathbf{t}}$ and the ground truth labels $\mathbf{t}$,

$$\mathcal{L}_{type} = \frac{1}{N_f} \sum_{j=1}^{N_f} \mathcal{H}(\mathbf{t}_j, \widehat{\mathbf{t}}_j) . \tag{5}$$

The total loss function is the sum of the *op.step* and *op.type* losses,

$$\mathcal{L}_{total} = \mathcal{L}_{step} + \mathcal{L}_{type} . \tag{6}$$

The model jointly learns to predict the per-face *op.type* and *op.step* labels of a CAD model given its B-Rep, with the *op.type* being conditioned on the *op.step*.

5. CC3D-Ops dataset

We introduce the *CC3D-Ops* dataset that contains $37k+$ B-Reps with the corresponding per-face *op.type* and *op.step* annotations. These labels were extracted using the Solidworks API [33]. The B-Reps and their corresponding annotations constitute an extension of the CC3D dataset [4]. While the Fusion360 dataset [40] contains a similar number of B-Reps ($35k+$) with the corresponding *op.type* labels, it does not provide *op.step* labels and it includes relatively simple CAD models. The proposed *CC3D-Ops* dataset comes with more complex models that are closer to real-world industrial challenges. In Figure 3, we illustrate the distribution of *op.step* number per model as a box plot for both Fusion360 and *CC3D-Ops* datasets. It can be clearly observed that the distribution of *CC3D-Ops* is more skewed towards a higher number of *op.steps* than the one of Fusion360. Specifically, $\sim 48\%$ of the Fusion360 models are made of only one *op.step* and $\sim 80\%$ of them are constructed by 3 or less *op.steps*. On the other hand, only $\sim 20\%$ of the *CC3D-Ops* models are built with a single *op.step* and $\sim 44\%$ of them with 3 or less *op.steps*. Moreover, the maximum number of *op.steps* per model, k_s, is 59 for Fusion360 and 262 for *CC3D-Ops*. Finally, the *CC3D-Ops* dataset introduces three new *op.types* to the eight present in Fusion360 which consists of, '*cut revolve side*', '*cut revolve end*', and '*others*'. More details about the dataset can be found in the supplementary material.

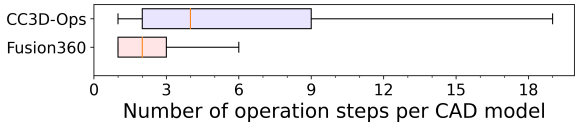


Figure 3: Boxplot of the number of *op.steps* per model in Fusion360 [40] and the proposed *CC3D-Ops* dataset.

6. Experiments

6.1. Experimental Setup

Input Features: The input features of *CADOps-Net* are face, edge and co-edge features $(\mathbf{F}, \mathbf{E}, \mathbf{C})$ extracted from the B-Rep, $\mathcal{B}$. Following [17], the face type (*e.g.* plane, cylinder, sphere) and area are encoded in a single vector. 3D points are further sampled on each face using the UV-grid of the B-Rep and encoded as described in [12]. These two features are concatenated and used as face features. The features of the B-Rep faces are then concatenated in a row-wise fashion to form the matrix $\mathbf{F}$. For edge features, a similar approach is taken by considering the type, convexity, closeness, length of the edge as in [17], and encoded sampled 3D points as done in [12]. The result is concatenated in an edge feature matrix $\mathbf{E}$. The co-edge features, $\mathbf{C}$, are simple flags to represent the direction of the corresponding edges [17].

Network Architecture: The input features are passed through a BRepNet backbone, $\mathbf{\Delta}$, with the same parameters as in [17] using the wing-edge kernel. The dimension of the face embedding, $\mathbf{f}^{\Delta}$, is $d_{emb} = 64$. These embeddings

are fed to an MLP followed by softmax, σ, to predict the *op.step*. The aggregation function used to compute the step embedding, $\mathbf{S}^{\mathcal{A}}$, is the average function. Each *op.step* embedding $\mathbf{s}^{\mathcal{A}}$ has the same dimension as $\mathbf{f}^{\Delta}$. The final face embedding, $\mathbf{f}^{\Delta} \oplus \mathbf{s}^{\mathcal{A}}$, are 128-dimensional. Lastly, the *op.type* is estimated by passing these embeddings through an MLP followed by softmax, ρ. In our experiments, the number of layers of the employed MLPs is 1.

Datasets: *CADOps-Net* is evaluated on the Fusion360 dataset [40] and the novel *CC3D-Ops* dataset described in Section 5. Note that in Fusion360, the *op.step* annotations were derived from the *op.type* annotations as they were implicitly provided. The train, validation, and test sets for the Fusion360 dataset are the same as in [17]. For the *CC3D-Ops* dataset, the splitting ratios are approximately 65%, 15%, and 20% for the train, validation, and test sets.

Training details: The training was conducted for 200 epochs with a batch size of 100 using an NVIDIA RTX $A6000$ GPU. Adam optimizer is employed with a learning rate of 0.001 and beta parameters of 0.9 and 0.99.

Metrics: The performance of the network is evaluated on *op.type* and *op.step* segmentation tasks. To evaluate the *op.type* segmentation, we use the same metrics as in [17], namely, the mean accuracy (mAcc) and the mean Intersection over Union (mIoU). Note that we do not consider the mIoU for evaluating the *op.step* as the labels represent membership sets rather than predefined classes. Furthermore, the consistency between the *op.type* and *op.step* predictions is considered. For this purpose, we group the sub-*op.types*, such that *'extrude end'* and *'extrude side'*, into a single *'extrude' op.type*. Similar grouping is done for *'revolve'*, *'cut extrude'*, and *'cut revolve'*. We define an *op.step* prediction as consistent if all its faces have the same *op.type* prediction. To evaluate this consistency, two metrics are computed: (1) the first one, R_C, quantifies the overall consistency as the ratio of consistent predicted *op.steps*; (2) the second one quantifies the amount of consistency of a model as $S_C = \sum_i \frac{\max(n_{(t_1,s_i)},\ldots,n_{(t_{k_t},s_i)})}{n_{s_i}}$ where n_{s_i} is the number of faces with *op.step* label s_i and $n_{(t_j,s_i)}$ the number of faces with *op.type* label t_j and *op.step* label s_i. We then compute mS_C as the average over all the models.

6.2. Results and Discussions

In this Section, we report and discuss the results of *CADOps-Net* compared to relevant SOTA works.

Qualitative Evaluation: In Figure 4, we illustrate the predictions obtained by *CADOps-Net* on five models from the *CC3D-Ops* dataset. More predictions are provided in the supplementary material. Despite the complexity of some models, it can be observed that most of the *op.type* predictions (left panel) were correct except for very few faces. On the other hand, the segmentation into *op.steps* (right panel) was more challenging for complex models (two last rows) as the segmentation into *op.steps* requires the model to learn the relationship between the faces of the B-Rep according to the construction history. Such aspect is more challenging to capture for complex models than the *op.types* which could be hypothetically learned from the geometry and topology of the B-Reps. This hypothesis is further discussed in the quantitative evaluation.

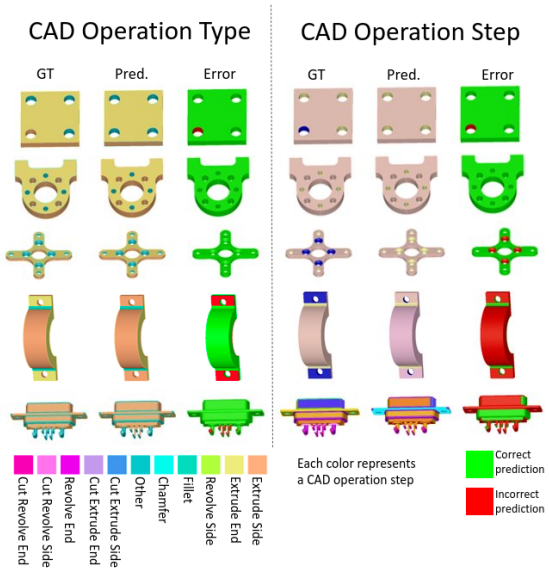
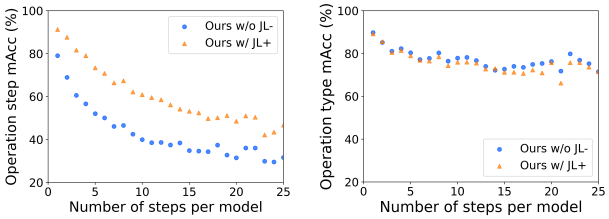


Figure 4: Sample predictions on five models from the *CC3D-Ops* dataset. (Left): The CAD operation type segmentation. (Right): The CAD operation step segmentation. For both tasks, the ground truth (GT) is shown in the left, the prediction (Pred.) in the middle, and the error (Error) in the right illustrating the correct/incorrect face predictions.



(a) CAD operation step mAcc (b) CAD operation type mAcc

Figure 5: Mean accuracy (mAcc) of CAD operation type and step segmentation *w.r.t* the number of steps per model on the *CC3D-Ops* dataset.

Quantitative Evaluation: In Table 1, we report the quantitative results of the proposed approach compared to baselines. For both tasks, *CADOps-Net* (*Ours w/ JL*$^+$) is com-

pared to the same model without the joint learning of *op.steps* and *op.types* (*Ours w/o JL*$^-$). In the latter, the *op.type* and *op.step* segmentation modules are trained independently. In the following, we first analyze the results for the segmentation into *op.steps* (column 5 of Table 1) and for the *op.type* segmentation (columns 3 and 4), then we discuss the consistency between the two types of predictions (columns 6 and 7).

	Model	*op.type*		*op.step*	*Consistency*	
		mAcc	mIoU	mAcc	R_C	mS_C
Fusion360	*CADNet* [5]	88.9	67.9	-	-	-
	UV-Net [12]	92.3	72.4	-	-	-
	BRepNet [17]	94.3	81.4	-	-	-
	Ours w/o JL$^-$	95.5	83.2	80.2	87.1	97.4
	Ours w/ JL$^+$	**95.9**	**84.2**	**82.5**	**93.3**	**98.7**
CC3D-Ops	*CADNet* [5]	57.5	26.9	-	-	-
	BRepNet [17]	71.4	35.9	-	-	-
	Ours w/o JL$^-$	**76.0**	43.0	48.4	40.7	82.7
	Ours w/ JL$^+$	75.0	**44.3**	**62.7**	**82.4**	**96.7**

Table 1: Results of the segmentation into CAD operation types and steps on the Fusion360 and *CC3D-Ops* datasets. All results are expressed as percentages. *Ours w/o JL*$^-$ denotes our method without joint learning. *Ours w/ JL*$^+$ refers to the proposed *CADOps-Net* with joint learning.

As previously mentioned, predicting *op.steps* is a much more challenging task than *op.types* especially for models with a large number of *op.steps*. While the joint learning leads to small improvements on the *op.step* mAcc metric on the Fusion360 dataset, significant improvements can be observed on the *CC3D-Ops* dataset results with an increase of $\sim$14%. This difference of results can be explained by the higher complexity of *CC3D-Ops* models compared to those of Fusion360. Figure 5a shows the mAcc of *op.step* segmentation related to the number of *op.steps* per model on the *CC3D-Ops* dataset. It can be observed that for models with less than 25 *op.steps*, representing over 96% of the *CC3D-Ops* dataset, *CADOps-Net* scores consistently and significantly better than without joint learning. These observations demonstrate the importance of the joint learning for *op.step* segmentation. However, in both cases there is a major decrease in the *op.step* segmentation mAcc as the number of steps per model increases. This is expected since the task becomes increasingly challenging as the number of *op.steps* becomes larger. Note that we did not compare our results to state-of-the-art (BRepNet [17], UV-Net [12], and CADNet[5]) on the task of *op.step* segmentation as their methods are not designed to predict arbitrary face labels.

In order to evaluate the *op.type* segmentation of *CADOps-Net*, the results are compared to SOTA results. On the Fusion360 dataset, we recorded slight improvements over [17], [12], and [5] in terms of mAcc. More significant improvements *w.r.t* [12] and [5] were obtained in terms of mIoU (more than 12% and 16%, respectively). On the *CC3D-Ops* dataset, our results clearly outperformed those of [17] and [5] on the two metrics. Furthermore, we compare *CADOps-Net* to the scenario where the joint learning is omitted. One interesting observation is that there is no significant difference between the two scenarios. The same observation holds in Figure 5b where we show the mAcc of *op.type* segmentation related to the number of *op.steps* per model. In contrast to the *op.step* segmentation, one can notice that the number of *op.steps* has a slight impact on the *op.type* mAcc. In other words, the *op.type* segmentation does not become more challenging when complex models with large number of construction steps are involved. Intuitively, it can be hypothesized that the *op.type* segmentation is more related to the geometry and topology of the B-Rep rather than its construction history.

The results on the consistency scores (R_C and mS_C) highlight the relevance of the joint learning approach. Despite relatively similar *op.type* and *op.step* mAcc scores on the Fusion360 dataset for *Ours w/ JL*$^+$ and *Ours w/o JL*$^-$, the joint learning approach produces more consistent results with an increase of $\sim$6% in R_C score. Similarly on the *CC3D-Ops* dataset, the predictions from *CADOps-Net* are significantly more consistent with an increase of $\sim$41% in R_C score and 14% in mS_C score. Therefore, the joint learning model is able to extract face features that contain consistent information for both the *op.type* and *op.step* segmentation labels. The consistency property is essential for the process of reverse engineering.

6.3. Ablation Study

	Agg. type	*op.type*		*op.step*
		mAcc	mIoU	mAcc
CC3D-Ops	*No agg.*	73.0	40.2	61.5
	Soft labels	73.4	40.0	59.7
	Sum	70.4	34.4	62.6
	Max	74.3	42.0	62.2
	Avg	**75.0**	**44.3**	**62.7**

Table 2: Ablation study on the aggregation function used in the joint learning of *CADOps-Net*. All results are expressed as percentages.

In order to provide a deeper insight into the joint learning approach, we conduct an ablation study on the aggregation function $\mathcal{A}$ of the face embeddings. Experiments are conducted with the following five scenarios: (1) the output face embeddings, $\mathbf{f}^{\Delta}$, from the BRepNet backbone are directly used to predict both the *op.type* and *op.step* without any aggregation (*No agg.*). (2) Another scenario concatenates the

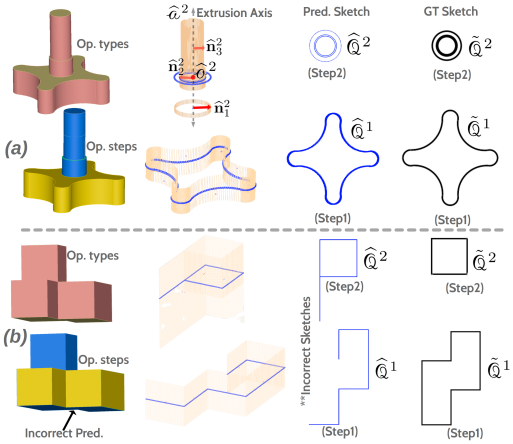


Figure 6: Sketch recovery from predicted CAD operation types (*op.types*) and steps (*op.steps*). *op.step* 1 and 2 are colored in yellow and blue, respectively. Figure 4 defines the color codes used for different *op.types*.

BRepNet face embeddings with the predicted soft labels of the *op.step* (*Soft labels*) again without any aggregation. (3) The last three scenarios focus on the type of aggregation function used to obtain the *op.step* embeddings, $\mathbf{S}^{\mathcal{A}}$, namely the maximum (*Max*), the average (*Avg*), and the sum of the embeddings combined with a softmax normalization (*Sum*). Table 2 shows the ablation results for both *op.type* and *op.step* segmentation tasks on the *CC3D-Ops* dataset. The results show that aggregating the face embeddings using an *Avg* pooling leads to slightly better overall performance.

6.4. CAD Sketch Recovery

Figure 6 illustrates preliminary results on how *CADOps-Net* predictions can be used to retrieve the CAD sketches. A sketch $\mathbb{Q}$ of a B-Rep $\mathcal{B}$ can be defined as a set of simple geometrical entities (*e.g.* straight lines, arcs). We consider a small subset of 20 models made of extrusions from the Fusion360 dataset. In the following, we describe the process for recovering the sketch corresponding to *op.step* 2 using the *CADOps-Net* predictions shown in Figure 6a. We first identify the faces for which the *op.type* was predicted as '*extrude side*'. Second, we cluster these faces according to their predicted *op.step*. Third, we store the face-normals $(\widehat{\mathbf{n}}_1^2, \ldots, \widehat{\mathbf{n}}_m^2)$ and sample UV-grid points on the faces. This allows to derive a common *axis of extrusion* $\widehat{a}$ and a *projection center* $\widehat{o}$. Finally, the predicted sketch $\widehat{\mathbb{Q}}^2$ is obtained by projecting the sampled points along $\widehat{a}$ (more details are in the supplementary material). Figure 6a and 6b show qualitative results of successful and failed sketch recoveries from correctly and incorrectly predicted *op.types*. These preliminary results on sketch recovery illustrate the relevance of *op.step* prediction in the context of 3D reverse engineering.

6.5. Limitations

In CAD modeling, different designers may opt for different design solutions for the same model. Consequently, the *op.type* and *op.step* segmentation is not necessarily unique. An example for which the *op.step* prediction is valid despite not matching the ground truth can be found in Figure 7a. The letters were predicted as part of the same *op.step*, which could be a valid design approach. However, these letters were extruded with separate *op.steps* in the ground truth. In Figure 7b, an example with valid predictions of *op.types* not matching the ground truth is depicted. Here, the hole in the center of the shape was predicted as created by a '*cut*' type operation, while being an '*extrude side*' in the ground truth. In general, CAD designers follow good practices so that the final model reflects the design intent [24]. However, different designers might have their own set of good practices, making it difficult for a learning-based model to capture all the different design intents.

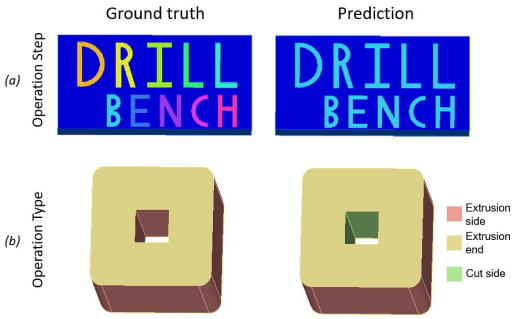


Figure 7: Failure cases of *CADOps-Net* for *op.step* segmentation in (a) and *op.type* segmentation in (b).

7. Conclusion

In this work, we have presented *CADOps-Net*, a neural network that jointly learns the CAD operation type and step segmentation of B-Rep faces. We showed that the joint learning strategy leads to significantly better results for the challenging task of CAD operation step segmentation, while achieving state-of-the-art results on the CAD operation type segmentation task. Moreover, we showed the potential of combining these two segmentations for recovering further useful information of the construction history such as sketches. Finally, we introduced the *CC3D-Ops* dataset and its operation type and step annotations. We believe that this dataset will help in further advancing research on CAD modeling thanks to the complexity of the CAD models it provides. As future work, we plan to investigate the ordering of the construction steps while maintaining various types of CAD operations. This would allow for the recovery of a more complete construction history.

References

[1] Sk Aziz Ali, Kerem Kahraman, Gerd Reis, and Didier Stricker. Rpsrnet: End-to-end trainable rigid point set registration network using barnes-hut 2d-tree representation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2

[2] Fatmir Azemi, Xhemajl Mehmeti, and Bekim Maloku. The importance of cad/cae systems in development of product design and process of optimization. In *2018 UBT International Conference*, Pristina, Kosovo, 2018. University for Business and Technology. 1

[3] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 2

[4] Kseniya Cherenkova, Djamila Aouada, and Gleb Gusev. Pvdeconv: Point-voxel deconvolution for autoencoding cad construction in 3d. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2741–2745, 2020. 2, 3, 5

[5] Andrew R Colligan, Trevor T Robinson, Declan C Nolan, Yang Hua, and Weijuan Cao. Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Computer-Aided Design*, 147:103226, 2022. 2, 3, 7

[6] 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):1–16, 2018. 1, 2

[7] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[8] Murilo Antonio Salomgo Garcia, Nelson Vogel, and Marcos de Sales Guerra Tsuzuki. New algorithm for converting a csg representation into a b-rep representation. In *Proceedings of the COBEM 2005: 18 th International Congress of Mechanical Engineering*, 2005. 1, 2

[9] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[10] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics*, 2019. 2

[11] Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *arXiv preprint arXiv:2203.13944*, 2022. 2

[12] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D.D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 3, 5, 7

[13] Byungchul Kim and Soonhung Han. Integration of history-based parametric translators using the automation apis. *International Journal of Product Lifecycle Management*, 2(1):18–29, 2007. 1

[14] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 3

[15] Philipp Krähenbühl and Vladlen Koltun. Parameter learning and convergent inference for dense random fields. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–513–III–521. JMLR.org, 2013. 4

[16] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 4

[17] Joseph G. Lambourne, Karl D.D. 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 (CVPR)*, pages 12773–12782, June 2021. 1, 2, 3, 4, 5, 6, 7

[18] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J. Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 4

[19] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[20] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[21] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2015. 2

[22] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 2

[23] MFCAD. A dataset of 3D CAD models with machining feature labels. https://github.com/hducg/MFCAD. Online: accessed 02-June-2022. 2, 3

[24] Jeffrey Otey, Pedro Company, 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):47–60, 2018. 8

[25] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[26] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NIPS'17, Red Hook, NY, USA, 2017. Curran Associates Inc. 2

[27] Aristides AG Requicha and Jarek R Rossignac. Solid modeling and beyond. *IEEE computer graphics and applications*, 12(5):31–44, 1992. 2

[28] Joachim Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. In *Computing and Visualization in Science*, 1997. 2

[29] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*, 2020. 2, 3

[30] Vadim Shapiro and Donald L. Vossler. Construction and optimization of csg representations. *Computer-Aided Design*, 23(1):4–20, 1991. 2

[31] 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*, pages 5515–5523, 2018. 1, 2

[32] E Solaberrieta, R Minguez, L Barrenetxea, E Sierra, and O Etxaniz. Computer-aided dental prostheses construction using reverse engineering. *Computer methods in biomechanics and biomedical engineering*, 17(12):1335–1346, 2014. 1

[33] Solidworks. 3D CAD Design Software. https://www.solidworks.com/. Online: accessed 02-June-2022. 5

[34] William B Thompson, Jonathan C Owen, HJ de St Germain, Stevan R Stark, and Thomas C Henderson. Feature-based reverse engineering of mechanical parts. *IEEE Transactions on robotics and automation*, 15(1):57–66, 1999. 1

[35] Stefano Tornincasa, Francesco Di Monaco, et al. The future and the evolution of cad. In *Proceedings of the 14th international research/expert conference: trends in the development of machinery and associated technology*, volume 1, pages 11–18. Citeseer, 2010. 1

[36] Michael Tovey. Drawing and cad in industrial design. *Design Studies*, 10(1):24–39, 1989. 1

[37] Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph Lambourne, Tolga Birdal, and Leonidas Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4

[38] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Trans. Graph.*, 36(4), jul 2017. 2

[39] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), oct 2019. 2

[40] 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 Trans. Graph.*, 40(4), jul 2021. 2, 3, 5, 6

[41] Qiaoyun Wu, Kai Xu, and Jun Wang. Constructing 3d csg models from 3d raw point clouds. In *Computer Graphics Forum*, volume 37, pages 221–232. Wiley Online Library, 2018. 2

[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, October 2021. 1, 2

[43] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[44] 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, 2

[45] Zhibo Zhang, Prakhar Jaiswal, and Rahul Rai. Featurenet: Machining feature recognition based on 3d convolution neural network. *Computer-Aided Design*, 101:12–22, 2018. 1