

Reconstructing editable prismatic CAD from rounded voxel models

JOSEPH G. LAMBOURNE, Autodesk Research, United Kingdom

KARL D.D. WILLIS, Autodesk Research, USA

PRADEEP KUMAR JAYARAMAN, Autodesk Research, Canada

LONGFEI ZHANG, Autodesk, China

ADITYA SANGHI, Autodesk Research, Canada

KAMAL RAHIMI MALEKSHAN, Autodesk Research, Canada

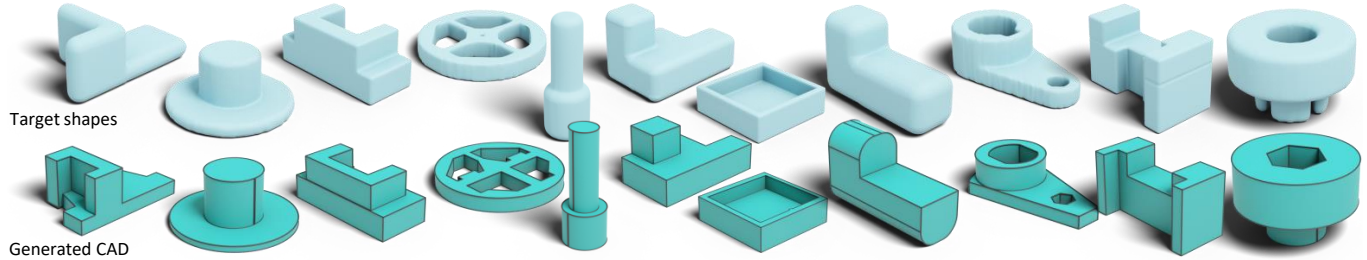


Fig. 1. Conversion of rounded voxel models to prismatic CAD.

Reverse Engineering a CAD shape from other representations is an important geometric processing step for many downstream applications. In this work, we introduce a novel neural network architecture to solve this challenging task and approximate a smoothed signed distance function with an editable, constrained, prismatic CAD model. During training, our method reconstructs the input geometry in the voxel space by decomposing the shape into a series of 2D profile images and 1D envelope functions. These can then be recombined in a differentiable way allowing a geometric loss function to be defined. During inference, we obtain the CAD data by first searching a database of 2D constrained sketches to find curves which approximate the profile images, then extrude them and use Boolean operations to build the final CAD model. Our method approximates the target shape more closely than other methods and outputs highly editable constrained parametric sketches which are compatible with existing CAD software.

CCS Concepts: • **Applied computing** → **Computer-aided design**; • **Computing methodologies** → **Neural networks**; **Parametric curve and surface models**.

Additional Key Words and Phrases: Computer aided design, CAD, reverse engineering, reconstruction, voxel

ACM Reference Format:

Joseph G. Lambourne, Karl D.D. Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. 2022. Reconstructing editable prismatic CAD from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers)*, December 6–9, 2022, Daegu, Republic of Korea. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3550469.3555424>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea

© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-9470-3/22/12...\$15.00

<https://doi.org/10.1145/3550469.3555424>

1 INTRODUCTION

Reverse engineering, the creation of Computer Aided Design (CAD) models which closely match some target geometry, is one of the most sought-after geometric modeling technologies and has been extensively studied using both traditional algorithms [Buonamici et al. 2018; Varady 2008] and machine learning approaches [Sharma et al. 2020; Uy et al. 2022; Wang et al. 2020]. Traditional techniques work well in cases where the target geometry is not noisy or otherwise distorted and primitive surfaces like planes and cylinders can be correctly identified and fitted [Benkő and Várady 2004; Li et al. 2011]. However, many sources of geometry provide only an approximate description of the desired shape. For example, noise in point clouds acquired by low cost scanners is often removed by smoothing the shape during surface reconstruction [Calakli and Taubin 2011] and shape manipulation approaches working on levelsets also result in rounding of the geometry [Sethian 1999]. As there are many effective methods to convert meshes and dense point clouds into signed distance functions stored in voxel grids [Bærentzen 2005; Sanchez et al. 2012], the ability to convert voxel models to CAD can be used as a stepping stone to reverse engineer these representations as well.

While free-form CAD data can readily be derived from rounded models by fitting splines to a quadrangulation of the surface [Bommes et al. 2009], the design of mechanical parts requires prismatic geometry composed of primitive surface types [Benkő et al. 2001]. Planar faces are important for the interfaces where components meet and cylindrical surfaces are required for holes and as parts of rotating mechanisms. In addition, shapes created from extrusions of connected lines and arcs are cheap to manufacture with low cost 2.5-axis Computer Numerical Control (CNC) machines. Most modern CAD software records these curves, along with the lengths and directions of extrusions, in a parametric recipe which designers can use to edit the model. The curve geometry is held in 2D sketches

along with constraints which maintain important aspects to design intent such as equality, symmetry, and perpendicularity. The designer can also define parameters which control distances between curves that can be varied to modify the geometry. By editing these parameters and replaying the recipe the shape of the final CAD model can be controlled.

Recently a number of methods have been proposed for the generation of CAD data as a sequence of geometry creation operations using transformer models [Ganin et al. 2021; Jayaraman et al. 2022; Para et al. 2021; Seff et al. 2022; Willis et al. 2021a; Wu et al. 2021]. These are all trained utilizing supervision from ground truth sequence data. As they do not incorporate a loss function which directly compares the generated geometry and the target shape, their ability to match target geometry is limited.

In this work we propose a differentiable pipeline which can reconstruct a target shape in terms of voxels and extract the parametric recipe simultaneously. The voxel model is created from a sequence of decoded profile images which are then extruded along the axes of the voxel grid and combined using Boolean operations. A geometric loss can then be defined which brings this voxel model closer to the target data. The recipe can then be extracted by matching the profile images with constrained parametric sketches which can be further fitted to better approximate the shape while maintaining design intent. We show that the CAD data reconstructed from the recipe contains fewer invalid solids and provides a better approximation to the target than DeepCAD [Wu et al. 2021], while having the additional benefit that the constraints and parameters in the sketches are available to designers for further editing.

2 RELATED WORK

We now review work from the literature related to the generation of editable CAD models.

Constructive Solid Geometry. A common approach to producing editable CAD models is to use a Constructive Solid Geometry (CSG) representation. With CSG, 3D shapes can be expressed as a tree of parametric primitives (e.g. cuboids, spheres, and cones) that are individually positioned and combined using Boolean operations (e.g. union, subtraction, intersection). This lightweight representation enables simple edits by adjusting the parameters of the primitives and their affine parameters, while also allowing more interesting applications with program synthesis [Du et al. 2018; Ellis et al. 2019; Nandi et al. 2018, 2020; Tian et al. 2019]. In the context of deep learning, this representation has been explored in several works such as [Chen et al. 2020; Deng et al. 2020; Kania et al. 2020; Sharma et al. 2018; Yu et al. 2022]. However, the primitives used in CSG are not as flexible as the extrusions of parametric sketches used by modern mechanical design tools. Consequently CSG shapes tend to be made from large number of primitives which are hard for users to edit to control the final geometry.

Engineering Sketch Generation. With the release of a large scale engineering sketch dataset [Seff et al. 2020], a number of learning-based approaches to engineering sketch generation have been explored [Ganin et al. 2021; Para et al. 2021; Seff et al. 2022; Willis et al.

2021a]. These works successfully frame engineering sketch generation as a sequence prediction task, where geometric primitives are sequentially predicted using a Transformer [Vaswani et al. 2017] model. Not addressed in these works is the generation of 3D parts compatible with parametric CAD.

CAD Generation. Numerous approaches for 3D shape generation have been proposed [Chaudhuri et al. 2020]. Related to our work are approaches that predict a sequence of CAD modeling operations from which a solid CAD model can be built [Willis et al. 2021b; Xu et al. 2021]. In particular, DeepCAD [Wu et al. 2021] is a generative model that learns from sequences of CAD modeling operations to produce editable CAD designs. In contrast to our approach, DeepCAD does not generate sketch constraints—a key factor aiding editability. CAD generation can also be achieved by predicting geometric entities and topological connections to form 3D shapes. PolyGen [Nash et al. 2020] introduced the idea of separating the sequential prediction of geometric points, from the topological connections that form n-gon meshes using pointer networks [Vinyals et al. 2015]. SolidGen [Jayaraman et al. 2022] applied a similar approach to the more difficult task of generating 3D shapes in the boundary representation (B-Rep) format commonly used in mechanical CAD. Missing from these sequence-based 3D generation approaches is a geometric loss suitable for reconstruction tasks.

CAD Reconstruction. The goal of CAD reconstruction is to *reverse engineer* a 3D shape, typically as a series of parametric primitives, given approximate input, such as a point cloud or freehand sketch. Smirnov et al. [2021] recover manifold 3D shapes from freehand sketch input by deforming Coons patch-based templates from set object categories. Given point cloud input, traditional approaches segment and then fit parametric primitives, such as planes, spheres, and cylinders, to the underlying point cloud [Schnabel et al. 2007]. Recent progress with learning-based approaches has addressed primitive segmentation [Yan et al. 2021] reconstruction of parametric curves [Wang et al. 2020] and surfaces [Guo et al. 2022; Li et al. 2019; Sharma et al. 2020]. Missing from these works is the sequence of CAD modeling operations that greatly enhances editability. Concurrent to our work, Point2Cyl [Uy et al. 2022] reconstructs a collection of extrusions which can be manually ordered and combined to build CAD shapes. An advantage of our approach is allowing for rounded or otherwise distorted shapes to be approximated by CAD data. As noisy or incomplete input data is often treated by smoothing, the ability to rebuild sharp prismatic shapes which approximate rounded geometry allows for a wider range of input.

Search and retrieval. An alternative approach to reconstruction is to search and retrieve an appropriate shape from an existing database. Parametric CAD models are well suited to retrieval because a range of different shapes can be returned from different input parameters. Schulz et al. [2017] address retrieval for parametric shapes in a collection by approximating their manifolds in the descriptor space with a set of primitives. Recent learning-based approaches actively deform the retrieved shape to more accurately match the input query [Uy et al. 2020, 2021]. In our work we utilize a search, retrieval and fitting procedure to identify parametric sketches with geometry close to a 2D target shape. The parameters in the retrieved

sketches can then be fitted to better approximate the shape of the target while maintaining the design intent encoded by the sketch constraints. This allows for novel workflows like interpolation between sketches providing smooth parametric variation in some regions and topological change in others.

3 DATA PREPARATION

A number of CAD related datasets have been released recently. Notably the ABC dataset [Koch et al. 2019] which provided 1,159,257 unique 3D CAD models, the SketchGraphs dataset [Seff et al. 2020] which contains 15 million constrained parametric 2D sketches, the Fusion Gallery Reconstruction dataset [Willis et al. 2021b] which provided 8,625 full sketch and extrude construction sequences along with sketch constraint information, and the DeepCAD dataset [Wu et al. 2021] which provided 127,267 unique sketch and extrude construction sequences without sketch constraints. In this work we utilize 3D data from DeepCAD and ABC, along with constrained sketches from SketchGraphs.

As the ABC dataset contains solids which require sophisticated free-form surface construction techniques like sweeps and lofts, we identify a subset of 52,503 models for which more than 95% of the surface area was consistent with extrusions aligned with the x, y or z axes. These are used as target geometry for some experiments and in the computation of the Fréchet inception distance (FID) metric.

3.1 Common extrusion combinations

Rather than attempting to produce arbitrary CAD modeling sequences, our approach focuses on the generations of B-Rep models which are built from pre-defined extrusion recipes, containing the axes along which profiles will be extruded, the order in which the extrusions are applied and whether each extrusion should be added or subtracted from the previous geometry. Most CAD software allows the designer to create sketches on existing planar faces of the solid being modeled, leading to some frequent patterns in the way extrusions are aligned and stacked. In this section we analyze the combinations of extrusions which designers employed when creating the models in the DeepCAD dataset [Wu et al. 2021] and find the most frequent patterns which are used as the basis for our extrusion recipes. After the deduplication procedure used in [Jayaraman et al. 2022], the dataset contains 127,267 files. Of these, just 358 contain two or more extrusions which are not orthogonal to one another, accounting for just 0.28% of the dataset. For this reason we focus only on the generation of solids built entirely from orthogonal extrusions. Without loss of generality, it is always possible to rotate orthogonal extrusion models such that the first extrusion will be created in the z direction. If a second extrusion direction exists, the model can then be rotated around the z axis so the x direction is aligned with the second extrusion direction. Transforming the models into this canonical pose allows 102,287 files (80.37%) of the dataset to be represented by the 18 combinations of extrusions shown in Appendix A.1 Figure 5. We see that single extrusions form the majority (56.7%) of the dataset. The next most frequent extrusion combination is a boss extruded from a sketch on the top face of the previous extrusion (see Figure 5b), which accounts for 5.24% of the models, followed by a cut which starts

from the top face of the previous extrusion (see Figure 5c) which accounts for 4.47% of the dataset.

3.2 Rounded shapes

Many processes which produce signed distance function models apply some smoothing to the shape as a kind of regularization. To allow our method to work well with these rounded shapes, we augment the dataset as follows. For each model we can create a signed distance voxel representation of the geometry in the canonical orientation described in Section 3.1 and scaled into the unit cube. We then generate rounded augmentations of each model by adding a constant to the signed distance function to offset the shape inward, re-initializing the distance function at the zero levelset [Sethian 1999] and reversing the process to offset outwards by the same amount. For each model in the dataset we add rounded augmentations with radii of 2.5%, 5.8%, 9.1%, 12.5% the size of the length of the bounding cube, corresponding to 1.6, 3.7, 5.8 and 8 voxels.

3.3 2D parametric sketch data

Constrained parametric sketches were obtained from the SketchGraphs dataset. As SketchGraphs contains collections of overlapping curves rather than closed profiles, the sketches were first processed to remove self-intersections. All intersecting curves were converted to construction geometry until the remaining geometry either formed closed loops or became disjoint. Disjoint sketches were discarded. An auto-constraining algorithm was then used to add as many constraints and driving parameters as possible without making the sketches over-constrained. This procedure was applied to 1,081,217 sketches from SketchGraphs and yielded 177,609 sketches which formed close loops and were suitable for downstream CAD operations. We then conducted an analysis of the sequences of lines and arcs present in profile loops in the DeepCAD dataset. The profile loops were clustered into groups which shared important geometric and topological properties as described in Appendix A.2. This analysis showed that 88% of profile loops in the DeepCAD dataset fall within the 50 largest groups. We then selected a small set of 49 constrained sketches which preserved the most common profile loop shapes over a wide range of parametric variations. Experiments showed that this small set of profile loops is able to approximate DeepCAD profiles surprisingly well, with an IoU of $93 \pm 11\%$ as described in Section 6.1. Experiments with more complex sets of profiles showed these sometimes resulted in extraneous details in the generated solids. The parameters of the selected sketches were then varied to give rise to 1690 shapes (more detail in Appendix A.2). The closed loops of 2D curves in the sketches were converted to SVG and rendered as binary images with resolution of 128x128. The fast marching method was used to convert the data to signed distance functions with negative values inside the object.

4 METHOD

Our method for reconstructing CAD shapes utilizes an autoencoder which encodes voxel models and decodes a prismatic shape as a collection of differentiable extrusions. The differentiable extrusions are created from 2D profile images representing the shapes to be extruded, along with 1D envelope arrays which define the start and

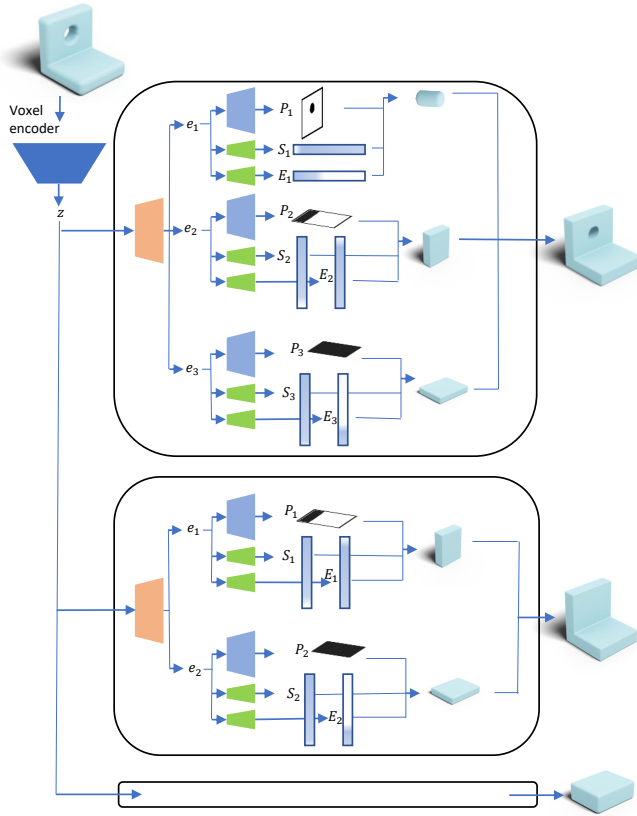


Fig. 2. The differentiable extrusion decoder architecture. The input voxel model is encoded to form an embedding z . A series of decoder modules then rebuild the shape using hard coded construction sequences. During training, only the ground truth decoder module is used, while at inference time the output of all decoder modules are evaluated and the one which best approximates the target shape is selected.

end positions of each extrusion. At inference time, the 2D images can be processed by a search, retrieval and fitting procedure, which yields parametric sketches suitable for extrusion by a CAD kernel, creating the final CAD model.

4.1 Differentiable extrusions

An overview of the architecture for decoding the differentiable extrusion voxel models is shown in Figure 2. The input is a signed distance function, represented as a voxel grid with resolution $64 \times 64 \times 64$. A standard voxel encoder as described in [Mescheder et al. 2019] can be used to convert this into a 128 length embedding vector z . This is passed to a series of decoder modules, each of which will attempt to interpret the shape in terms of a predefined extrusion sequence recipe as described in Section 3.1.

For a decoder module with a sequence of n extrusions, the voxel embedding, z , is split into extrusion embeddings e_0, e_1, \dots, e_{n-1} . This is done by a series of linear layers separated by ReLU non-linearities (see Appendix A.4). The extrusion embeddings can then be passed to a shared 2D deconvolution image decoder (see Appendix A.6) which

will generate images with resolution 128×128 , representing each profile, P_0, P_1, \dots, P_{n-1} . The decoded pixel values will be negative for pixels inside the profile and positive outside it.

In addition to the locations of the start and end of the extrusions needs to be found. To achieve this we predict 1D arrays of envelope values which can be used to limit the extrusions in a differentiable way. If the i th extrusion does not share a common start or end plane with others in the model, we decode two arrays of length 64 to represent its limits. The start array, S_i will contain positive values below the extrusions start plane and negative values above it, while the end array, E_i will be negative below the end plane and positive above it. These arrays are decoded from the extrusion embeddings using a 1D deconvolution network as described in Appendix A.8. For some frequent cases where two extrusions share a common plane, a single start/end array can be decoded and shared by both of them. This is explained in more detail in Appendix A.3. Taking the max of S_i and E_i defines an envelope which is negative only for the parts of the i th extrusion which we want to appear in the model.

With this information we can construct a $64 \times 64 \times 64$ voxel grid representation of the model in a differentiable way. For each of the n extrusions, the profile P_i is first downsampled as described in Appendix A.7, and then duplicated along the appropriate dimension of a 3D tensor as defined by the hard coded recipe in the decoder module. The envelope is then applied by taking the max of each slice of the tensor with its corresponding value in the envelope array. Finally the extrusions are combined with the Boolean operations in the recipe, using the min function for union, max for intersection and multiplying by -1 to form the complement. This gives rise to a signed function, ϕ of logits which, if passed through a sigmoid, would be the probability of a voxel being *outside* the model. Adopting this convention makes ϕ negative inside the object, allowing the shape to be visualized directly from the logits using marching cubes. Note that ϕ is not a distance function as the Eikonal equation is not satisfied as we move away from the zero level set.

4.2 Loss functions

The model is trained using two loss functions, a supervised loss which requires knowledge of the CAD construction sequence, and an unsupervised loss which does not require any construction sequence information. Both the supervised and unsupervised contributions to the loss are computed using a binary cross entropy "with logits" loss which includes the sigmoid function allowing it to operate directly on the predicted logits

$$BCE(x, y) = -\text{mean} [y \ln(\sigma(x)) + (1 - y) \ln(1 - \sigma(x))] \quad (1)$$

where x is the tensor of predicted logits, y is the binary target value, $\sigma()$ is a sigmoid function and the mean is over all the elements of the tensor. For the unsupervised loss function, the target voxel values \hat{T} are derived from the CAD data before any rounded augmentation is applied. \hat{T} is 1 for voxels outside the object and 0 inside the object. The unsupervised loss for the differentiable voxel model is then

$$L_{vox} = BCE(\phi, \hat{T}) \quad (2)$$

The supervised loss is computed using the ground truth 2D binary profile images, \hat{P}_i and 1D binary arrays, \hat{S}_i and \hat{E}_i , corresponding to the start and end envelope arrays. Following the convention above,

these tensors have values of 1 outside the object and 0 inside. We compute a binary cross entropy loss, averaged over the n extrusions utilized by the decoder module as

$$L_{profile} = \frac{1}{n} \sum_i BCE(P_i, \hat{P}_i) \quad (3)$$

$$L_{start} = \frac{1}{n} \sum_i BCE(S_i, \hat{S}_i) \quad (4)$$

$$L_{end} = \frac{1}{n} \sum_i BCE(E_i, \hat{E}_i) \quad (5)$$

The final loss is then computed as

$$L = L_{vox} + L_{profile} + L_{start} + L_{end} \quad (6)$$

4.3 Decoder module selection

During supervised training, the ground truth decoder module for each training example is known. At inference time this information is not available, so in order to make this selection we compare the input voxel model with the decoded differentiable extrusion model from each decoder module. The unsupervised binary cross entropy loss function, L_{vox} , can be evaluated for each of the available decoder modules and the module which results in the smallest loss is utilized. This procedure is also extended to allow regeneration of CAD models where the first extrusion is not in the z direction. The input voxel grid can be rotated 90 degrees around the x and y axes before the data is passed to the network. The differentiable extrusion model is then created for each decoder module with the target shape encoded in all three orientations. Then the best orientation and decoder module combination can be selected. Finally the regenerated CAD model is transformed back into its original pose.

4.4 Retrieving sketches and rebuilding CAD

At inference time the 2D profile images and 1D envelope arrays can be used to create a parametric recipe suitable for use in a CAD system. The profile images are converted to CAD profiles using a search procedure in an embedding space defined by a 2D autoencoder. This is built from the CNN encoder and deconvolution decoder as described in Appendix A.5 and A.6. As in the 3D case, the input images are signed distance functions and the model is trained using the loss from Equation 3. Once trained, the 2D autoencoder can be used to generate embeddings for each variation of a collection of constrained parametric sketches. The 2D profile images created by the 3D decoder network can then be converted to binary images by selecting the pixels with values lower than zero. The resulting shapes can be separated into connected components, each of which will be converted into a single CAD profile. The shape used to search for the outer loop of each component can be found by filling any internal holes and the inner loops extracted as the difference between the original and filled shape. For each loop extracted in this way, we can zoom in on a square around its binary mask. As the 2D autoencoder is trained using signed distance functions as input, the zoomed images representing single loops need to be converted to signed distance functions using the fast marching method [Sethian 1999]. The search then finds the parametric sketch variation with

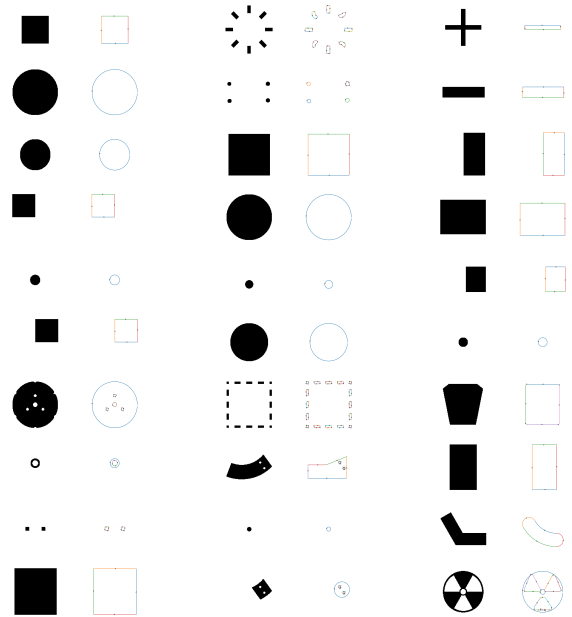


Fig. 3. Randomly selected retrieval results for profiles from the DeepCAD dataset. The target shapes are shown in black and the retrieved profile loops shown as colored curves. As a large fraction of the dataset is created from rectangular and circular loops, many of the shapes are very well approximated even with a small search set.

the embedding closest, in terms of euclidean distance, to the embedding of the signed distance query image. Once the sketch has been retrieved, its parameters can be fitted to better approximate the query shape. In addition to the internal parameters of the parametric sketch, the translation in x and y and sketch scale factor can also be optimized. To rebuild the CAD model we also need the positions of the start and end planes for each extrusion. These are determined by finding the positions of zero crossings of the extrusion’s start and end arrays. If multiple zero crossings exist, the planes are chosen such that the length of the extrusion is maximized. A CAD kernel can then be used to generate a B-Rep extrusion of the retrieved profile. The Boolean operations from the decoder module’s recipe are then used to combine the extrusions to create the CAD model.

5 METRICS

The following metrics are used to evaluate the performance of the generation technique.

Solid model validity. The validity of a solid model is critical to its usability in downstream processes. Solid models may contain problems which are not visible when the model is rendered, but cause the failure of downstream modeling operations. For this reason we test validity using the Open Cascade BRepCheck_Analyzer.IsValid() function as in code provided with [Wu et al. 2021]. The valid ratio is measured as the fraction of voxel models which generate solid data passing the Open Cascade validity test. Models which fail the validity test are not included in downstream metrics. Both the IoU

and FID metrics described below are computed using a voxelized version of the generated CAD. This cannot be generated reliably unless the solid is valid and watertight.

Intersection over Union. The intersection over union metric, also known as Jaccard index, provides an intuitive way to understand how well the regenerated model represents the shape of the target. The CAD models are first triangulated and then voxelized, taking care to correctly account for scaling and translation. The IoU can then be computed based on the resulting binary voxel grids. IoU measurements are always made with respect to the ground truth voxel models before any rounded augmentation has been applied. Because our approach canonicalizes the orientation of the voxel models, our voxel embeddings do not contain information about which orientation the final model needs to be constructed in. For this reason DeepCAD often creates shapes which look approximately correct, but are reconstructed in the wrong orientation. To ensure a fair comparison with DeepCAD, we rotate the target voxel model by all 24 non-mirror symmetries of cube and pick the biggest value when computing the DeepCAD IoU.

Fréchet inception distance. The Fréchet inception distance (FID) [Heusel et al. 2017] is a common metric used to measure quality and diversity of generated shapes as compared to a set of ground truth shapes. The metric is computed by measuring the difference in the distribution of activations obtained from a pre-trained model for generated shapes and ground truth shapes. In our case, we compute the metric using embeddings from a 3D voxel autoencoder, trained on 36,752 models from the ABC dataset which were identified as being consistent with axis aligned extrusions. Both the ground truth and generated data distributions were then created by voxelizing solid models and using the encoder of this autoencoder to produce the embeddings. For experiments with the DeepCAD dataset, the ground truth solids were created by reconstructing 5,000 randomly selected examples from the DeepCAD training set. For experiments on ABC the 5,000 ground truth solids were randomly selected from the same distribution of axis aligned models from ABC. These were not used for training or as target geometry in the experiments.

6 EXPERIMENTS

6.1 2D profile fitting

In this section we examine the performance of our 2D search retrieval and fitting technique at reproducing the shapes found in the profiles from 3,379 randomly selected models from the DeepCAD dataset. The 2D autoencoder was trained on 128,400 2D profiles generated by cutting axis aligned slices through 5,422 models in the ABC dataset. For each extrusion in the DeepCAD models, we create binary images from the union of the extruded profiles. The search and retrieval procedure described in Section 4.4 is then used to approximate the loops of the profiles using the 1,690 constrained parametric sketch variations created by the procedure described in Section 3.3. The IoU between the target profiles and the recovered sketches is $93 \pm 11\%$ which is surprisingly high given the very small number of sketches over which the search is conducted. We also investigated using the binary mask as input to the search network rather than a signed distance function. This results in a substantially

Dataset	Method	Valid ratio (\uparrow)	IoU (\uparrow)	FID (\downarrow)
DeepCAD	DC-P	69.47%	$47 \pm 11\%$	1.72e+6
	DC-V	71.46%	$48 \pm 12\%$	1.28e+6
	Ours	83.84%	$79 \pm 15\%$	1.3e+6
ABC	DC-P	56.99%	$51 \pm 7\%$	1.73e+6
	DC-V	51.66%	$50 \pm 6\%$	7.23e+5
	Ours	81.46%	$74 \pm 13\%$	2.43e+5

Table 1. Performance of the networks. DC-P is DeepCAD with the official point cloud encoder generating the embeddings. DC-V uses our voxel encoder to generate the embeddings. The mean and standard deviation of the IoU values are shown. The wide distribution reflects the fact that some models are better approximated than others.

lowers the IoU to $84 \pm 21\%$. We speculate that using a signed distance function breaks the translation invariance in the CNN encoder architecture in a similar way to the solution suggested by [Liu et al. 2018]. Figure 3 shows some randomly selected examples of target 2D shapes and the retrieved profiles. As 41.53% of DeepCAD profile loops are circles and 25.97% are rectangles, we see that the successful recovery of these simple shapes provides an excellent foundation for the search based approach.

6.2 3D reconstruction

Here we compare the performance of our reconstruction technique with DeepCAD. As our method can only generate the extrusion combinations for 80.37% of the models in the DeepCAD dataset, we use our own train/validation/test split of 70%/15%/15%. The official DeepCAD split provides only 5% of the models in the test set and does not account for duplicates. As target geometry, we utilize solids from the DeepCAD test set and from ABC. The ABC data is chosen such that can be built from extrusions, as described in Section 3.

Our network is trained on the DeepCAD dataset as described in Appendix A.9. We reconstruct CAD for 6,338 voxel models from DeepCAD and 7,876 models from ABC. For comparison with DeepCAD we use their pre-trained sequence encoder to generate sequence embeddings for our training set. We then train two encoders to replicate these embeddings. The original DeepCAD point cloud model is trained using point clouds extracted from triangulations of our voxels models, including the rounded augmentations. We also utilize our voxel encoder to consume the same voxel models as our network and produce the embeddings required for DeepCAD sequence generation. Both encoders are trained with MSE loss as in the official code from [Wu et al. 2021].

The results are shown in table 1. Our technique achieves the highest valid ratios of 83.84% on the DeepCAD dataset and 81.46% on ABC. DeepCAD has a tendency to create self-intersecting profile loops which result in invalid solids. In IoU, our method also outperforms DeepCAD. We observe that DeepCAD often creates plausible models with approximately the correct shape, but these do not fit as closely to the target shape as ours. We also record the IoU values for the differentiable voxel model produced by our approach. On the DeepCAD dataset this gives a value of $88 \pm 12\%$ and for

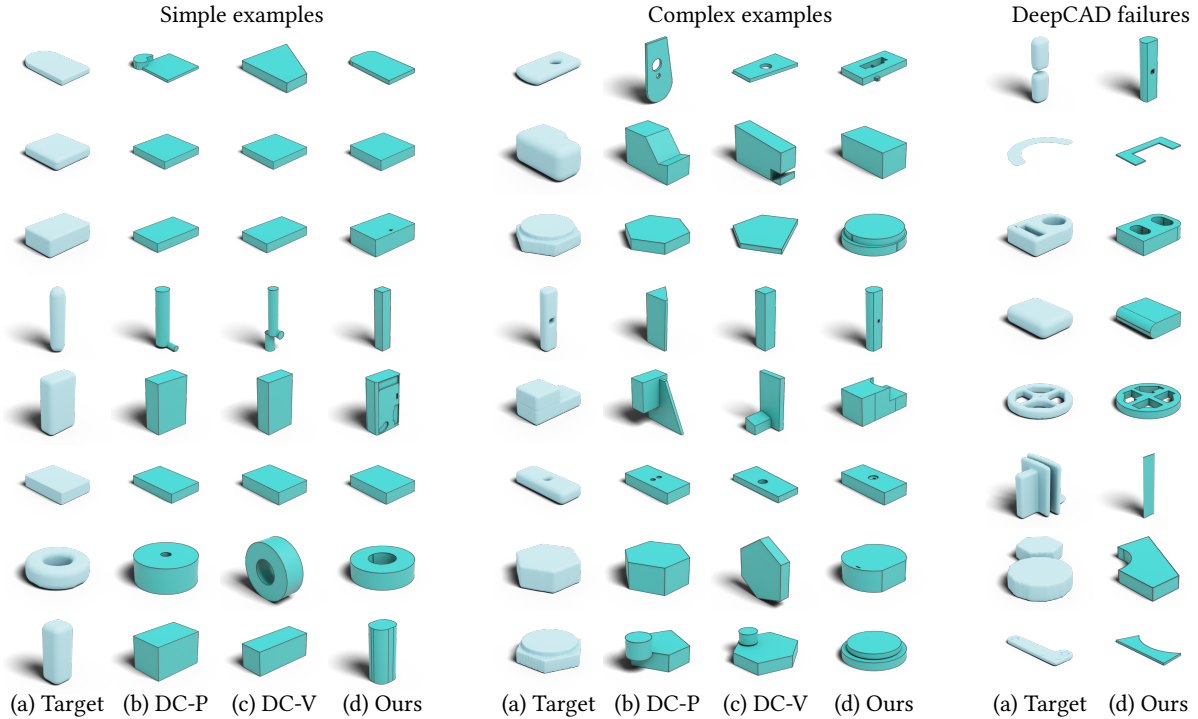


Fig. 4. Randomly chosen reconstructed models. On the left we show models which all methods could create, including blocks and cylinders. In the middle we show more complex examples. We exclude blocks and cylinders and then randomly sample from the remaining reconstructed results. On the right we show examples where DeepCAD failed to generate valid solids, but our method was able to do so successfully. (a) Target voxel model. (b) DeepCAD using the point cloud encoder. (c) DeepCAD using the voxel encoder. (d) Ours

ABC $81 \pm 13\%$. This shows that between 7% and 9% of the mismatch between the target and final model results from CAD conversion. This is approximately consistent with the 2D search and retrieval results where we observe the parametric sketches can approximate the ground truth profiles with a mean IoU of 93%.

In FID, the DeepCAD model marginally outperforms ours on the DeepCAD dataset. We observe that DeepCAD tends to create data which is very close to the original dataset, with little adaptation to the target shape as shown quantitatively by the IoU metric. On the ABC dataset our method outperforms DeepCAD in FID, showing it is better able to generalize to different data distributions.

To show results qualitatively we randomly sample models from the DeepCAD experiments for display in Figure 4. Only 51% of the voxel models were successfully regenerated by all three methods. The DeepCAD dataset contains a large number of blocks and cylinders, so when we randomly sample examples from the successfully regenerated CAD, (see left of Figure 4), the results are dominated by these simple shapes. To show results of a wider variety of geometries we filter out blocks and cylinders, and then randomly sample the models in the middle of Figure 4. Our technique is able to generate valid solids in cases where the DeepCAD result was invalid. We show random samples from these on the right hand side of Figure 4. We show similar results using ABC as the target models in the appendix Figure 10. We observe that as the target geometry become more complex, our technique discovers simplified shapes

which approximate it, while the DeepCAD results become random collections of simple extrusions.

7 CONCLUSIONS AND LIMITATIONS

This paper described a method for reconstructing CAD geometry from rounded voxel models. It allows prismatic reconstruction of axis aligned extrusion models from the ABC and DeepCAD dataset with state of the art IoU between the reconstructed geometry and the original model before rounding is added. The method is limited in that only a small set of pre-defined combinations of extrusions can be regenerated. In practice we observe that a wide variety of useful shapes can be modeled using the small set of extrusion combinations supported. The search and retrieval strategy employed to recover the CAD geometry is also limited in that it can only recover sketches from a predefined search set. While this narrows the range of topologies of profile loops which can be created, it works relatively well with the simple shapes in the available data. The main limitation of the approach is that each profile loop is converted to CAD geometry independently. This allows misalignment between curves and causes mismatched and incongruent profiles to be combined in a single model. For example the right hand model in Figure 1 shows a case where a hexagon was retrieved in place of a circular hole. In future work we plan to refine the CAD geometry conversion technique to consider the association between loops and generate geometry consistent with the entire solid.

REFERENCES

- J Andreas Bærentzen. 2005. Robust generation of signed distance fields from triangle meshes. In *Fourth International Workshop on Volume Graphics, 2005*. IEEE, IEEE, 167–239.
- Pál Benkő, Ralph R. Martin, and Tamás Várady. 2001. Algorithms for reverse engineering boundary representation models. *Computer-Aided Design* 33, 11 (2001), 839–851. [https://doi.org/10.1016/S0010-4485\(01\)00100-2](https://doi.org/10.1016/S0010-4485(01)00100-2)
- Pál Benkő and Tamás Várady. 2004. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design* 36, 6 (2004), 511–523. [https://doi.org/10.1016/S0010-4485\(03\)00159-3](https://doi.org/10.1016/S0010-4485(03)00159-3)
- David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-Integer Quadrangulation. *ACM Trans. Graph.* 28, 3, Article 77 (jul 2009), 10 pages. <https://doi.org/10.1145/1531326.1531383>
- Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, and Yary Volpe. 2018. Reverse engineering modeling methods and tools: a survey. *Computer-Aided Design and Applications* 15, 3 (2018), 443–464. <https://doi.org/10.1080/16864360.2017.1397894> arXiv:<https://doi.org/10.1080/16864360.2017.1397894>
- F. Calakli and Gabriel Taubin. 2011. SSD: Smooth Signed Distance Surface Reconstruction. *Computer Graphics Forum* 30 (11 2011), 1993 – 2002. <https://doi.org/10.1111/j.1467-8659.2011.02058.x>
- Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. 2020. Learning generative models of 3D structures. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, Wiley, 643–666.
- Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 45–54.
- Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. 2020. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 31–44.
- E. W. Dijkstra. 1959. A Note on Two Problems in Connexion with Graphs. *NUMERISCHE MATHEMATIK* 1, 1 (1959), 269–271.
- Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. 2018. Inversecsg: Automatic conversion of 3d models to csg trees. *Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* 37, 6 (2018), 1–16.
- Kevin Ellis, Maxwell Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. 2019. Write, execute, assess: Program synthesis with a repl. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, Article 822, 10 pages.
- Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. 2021. Computer-aided design as language. In *Advances in Neural Information Processing Systems (NeurIPS)*. Advances in Neural Information Processing Systems (NeurIPS).
- Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. 2022. ComplexGen: CAD Reconstruction by B-Rep Chain Complex Generation. *ACM Transactions on Graphics (TOG)* 41 (2022), 1 – 18.
- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).
- Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. 2022. SolidGen: An Autoregressive Model for Direct B-rep Synthesis. *arXiv preprint arXiv:2203.13944* (2022).
- Kacper Kania, Maciej Zięba, and Tomasz Kajdanowicz. 2020. UCSG-Net—Unsupervised Discovering of Constructive Solid Geometry Tree. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vancouver, BC, Canada) (NIPS'20). Curran Associates Inc., Red Hook, NY, USA, Article 736, 11 pages.
- Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9593–9603.
- Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, L. Yi, and Leonidas J. Guibas. 2019. Supervised Fitting of Geometric Primitives to 3D Point Clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 2647–2655.
- Yanyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. 2011. GlobFit: Consistently Fitting Primitives by Discovering Global Relations. *ACM Transactions on Graphics* 30, 4, Article 52 (2011), 12 pages.
- Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. 2018. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (Montréal, Canada) (NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 9628–9639.
- Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Chandrakana Nandi, James R Wilcox, Pavel Panchekha, Taylor Blau, Dan Grossman, and Zachary Tatlock. 2018. Functional programming for compiling and decompiling computer-aided design. *Proceedings of the ACM on Programming Languages* 2, ICFP (2018), 1–31.
- Chandrakana Nandi, Max Willsey, Adam Anderson, James R. Wilcox, Eva Darulova, Dan Grossman, and Zachary Tatlock. 2020. Synthesizing Structured CAD Models with Equality Saturation and Inverse Transformations. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. 31–44.
- Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. 2020. PolyGen: An Autoregressive Generative Model of 3D Meshes. *ICML* (2020).
- John A. Nelder and Roger Mead. 1965. A simplex method for function minimization. *Computer Journal* 7 (1965), 308–313.
- Wamiq Reyaz Para, Shariq Farooq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas Guibas, and Peter Wonka. 2021. SketchGen: Generating Constrained CAD Sketches. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Mathieu Sanchez, Oleg Fryazinov, and Alexander Pasko. 2012. Efficient evaluation of continuous signed distance to a polygonal mesh. In *Proceedings of the 28th Spring Conference on Computer Graphics*. Association for Computing Machinery, New York, NY, USA, 101–108.
- Ruwen Schnabel, Roland Wahl, and R. Klein. 2007. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum* 26 (2007).
- Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. 2017. Retrieval on Parametric Shape Collections. *ACM Trans. Graph.* 36, 1, Article 11 (Jan. 2017), 14 pages. <https://doi.org/10.1145/2983618>
- Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. 2020. SketchGraphs: A Large-Scale Dataset for Modeling Relational Geometry in Computer-Aided Design. In *ICML 2020 Workshop on Object-Oriented Learning*.
- Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P. Adams. 2022. Vitruvius: A Generative Model of Parametric CAD Sketches. In *International Conference on Learning Representations (ICLR)*.
- J.A. Sethian. 1999. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press. <https://books.google.co.uk/books?id=ErpOoynE4dIC>
- Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2018. CSGNet: Neural Shape Parser for Constructive Solid Geometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomir Měch. 2020. ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds. arXiv:2003.12181 [cs.CV]
- Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research* 12 (2011), 2539–2561. <http://dl.acm.org/citation.cfm?id=2078187>
- Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. 2021. Learning Manifold Patch-Based Representations of Man-Made Shapes. In *International Conference on Learning Representations (ICLR)*.
- Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. 2019. Learning to Infer and Execute 3D Shape Programs. In *International Conference on Learning Representations (ICLR)*.
- Mikaela Angelina Uy, Yen-yu Chang, Minhyuk Sung, Purvi Goel, Joseph Lambourne, Tolga Birdal, and Leonidas Guibas. 2022. Point2Cyl: Reverse Engineering 3D Objects from Point Clouds to Extrusion Cylinders. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. 2020. Deformation-aware 3d model embedding and retrieval. In *European Conference on Computer Vision (ECCV)*. Springer, 397–413.
- Mikaela Angelina Uy, Vladimir G. Kim, Minhyuk Sung, Noam Aigerman, Siddhartha Chaudhuri, and Leonidas Guibas. 2021. Joint Learning of 3D Shape Retrieval and Deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Tamas Várady. 2008. Automatic Procedures to Create CAD Models from Measured Data. *Computer-Aided Design and Applications* 5, 5 (2008), 577–588. <https://doi.org/10.3722/cadaps.2008.577-588> arXiv:<https://www.tandfonline.com/doi/pdf/10.3722/cadaps.2008.577-588>
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)* (2017), 5998–6008.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in Neural Information Processing Systems (NeurIPS)* (2015).
- Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. 2020. PIE-NET: Parametric Inference of Point Cloud Edges. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. Curran Associates, Inc., 20167–20178.
- Karl D. D. Willis, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Hang Chu, and Yewen Pu. 2021a. Engineering Sketch Generation for Computer-Aided Design. In *The 1st Workshop on Sketch-Oriented Deep Learning (SketchDL)*, CVPR 2021.
- Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021b. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).

- Rundi Wu, Chang Xiao, and Changxi Zheng. 2021. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D. D. Willis, and Daniel Ritchie. 2021. Inferring CAD Modeling Sequences Using Zone Graphs. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)*, 6058–6066.
- Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qi-Xing Huang. 2021. HPNet: Deep Primitive Segmentation Using Hybrid Representations. *2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021)*, 2733–2742.
- Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. 2022. CAPRI-Net: Learning Compact CAD Shapes with Adaptive Primitive Assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11768–11778.

A APPENDIX

A.1 Combinations of extrusions

Figure 5 shows the 18 combinations of extrusion alignments, shared start/end planes and Boolean operations used by decoder modules in this work. Note that this figure shows extrusions of rectangular and circular profiles as illustrations, while the real dataset utilizes more complex profile shapes. For pairs of extrusions which have the same extrusion axis and share a common start or end plane (shown in orange in Figure 5), the shared plane is encoded as an additional constraint as described in Appendix A.3.

A.2 DeepCAD profile analysis and selection

We performed the following analysis on the 2D profiles from DeepCAD. Each profile loop was converted into a graph with curves as nodes and edges linking adjacent curves. Node attributes were defined based on curve type and whether lines were aligned with the coordinate system axes. Horizontal and vertical lines are common in CAD profiles and can be considered as different to lines with arbitrary directions. Edge attributes were used to indicate whether adjacent curves met with tangent continuity. We then compute the Weisfeiler Lehman graph hash [Shervashidze et al. 2011] for these graphs. The DeepCAD profile loops can then be organized into groups which have the same hash. We select constrained parametric sketches which retain the same graph hash as the largest groups over a wide range of parametric variations. In Figure 6 we show 12 of the constrained parametric sketches used in this work, ranked by the size of the group of DeepCAD profiles with the same graph hash. We see that circles and rectangles dominate the dataset. The other shapes are frequently produced by other sketch generation works [Ganin et al. 2021; Para et al. 2021; Seff et al. 2020, 2022]. As these sketches are parametric, their parameters can be varied to form a family of geometries. For example the L-shaped sketch in Figure 6g can be manipulated to build L-shaped profiles with any aspect ratio, while the constraints ensure that all lines remain vertical and horizontal as in the image. To generate parametric variations of our template sketches we change the parameters in increments, flooding out from the initial parameter values using a priority queue as in Dijkstra’s algorithm [Dijkstra 1959]. Parameter values which result in self-intersections or constraint solver failures do not propagate to further nodes. The algorithm stops when the priority queue becomes empty or a set number of variations have been successfully generated. Using this procedure we requested 1000 variations of the rectangular profile and 30 variations of all others except the circles (which have no degrees of freedom beyond center and scale). This results in 1690 variations which are then used in the search, retrieval and fitting procedure described in section 4.4.

A.3 Sharing envelope arrays

When creating CAD models from multiple extrusions which share a common plane, it is important that the extrusions start/end exactly at that plane. Small differences between the end of one extrusion and the start of the next can cause a small gap in the model which is problematic for downstream workflows.

In this work we carefully design the decoder modules to constrain shared planes to have identical positions for the planes shown in

orange in Figure 5. This is done by decoding a single envelope array for the height of the shared plane, and using these values to build the envelopes for each extrusion abutting the plane.

Figure 7 shows an example of the envelope arrays for two abutting extrusions. We use a single envelope decoder in the decoder module which decodes the values of E_1 . The values of S_2 are then computed as $S_2 = -E_1$.

A.4 Operation decoder

The linear layers which decode the voxel embedding, z , into the individual embeddings for each extrusion have sizes: 128, 512, 512, $128n$, where n is the number of extrusions. We use a ReLU non-linearity after each linear layer. Following the last layer, the output tensor is split into the n extrusion embeddings e_0, e_1, \dots, e_{n-1} , each with size 128.

A.5 2D encoder

The 2D CNN encoder used 6 Conv2d layers with leaky ReLU non-linearity. The kernel size was 4 and the stride was 2. The channel sizes were 1, 64, 64, 128, 256, 512, 64.

A.6 2D decoder

The decoder used 7 ConvTranspose2d layers, the first of which was a 1×1 conv. After this the kernel size was 4, the stride was 2. The channel sizes were 64, 512, 64, 64, 32, 32, 16, 1. When decoding the 128 length extrusion embeddings, e_i , a linear layer was used to map the size down to the length 64 image embeddings used by the decoder.

A.7 2D image downsampler

An advantage of the suggested architecture is that the 2D profile images, generated by the image decoders, can be created with a higher resolution than the voxel grid in which the differentiable extrusions are created. When working with data where the CAD construction sequence is known, the supervised component of the loss can also operate on the images at this higher resolution. In this work the decoded profile images have resolution 128×128 and the differentiable extrusions are created with a voxel grid of resolution $64 \times 64 \times 64$. Before a profile is extruded it is downsampled using a Conv2d layer with fixed weights. The kernel size was 3, stride was 2 and padding 1. The weights were initialized to $1/3$ and not passed to the optimizer.

A.8 Envelope decoder

The 1D envelope decoders are analogous to the 2D image decoders used to create profile images. They are implemented using 6 ConvTranspose1d layers with ReLU non-linearities between them. The channel sizes were 128, 512, 64, 64, 32, 32, 1. The first layer had kernel size 1, while subsequent layers used kernel size 4. The first two layers had a stride of 1 and subsequent layers had stride 2.

A.9 Training

Both the 2D and 3D autoencoders were trained with the Adam optimizer and a learning rate of $2e-4$. The 2D autoencoder was trained on 4 Quadro RTX 6000 GPUs and took 1 day 14h to converge. The

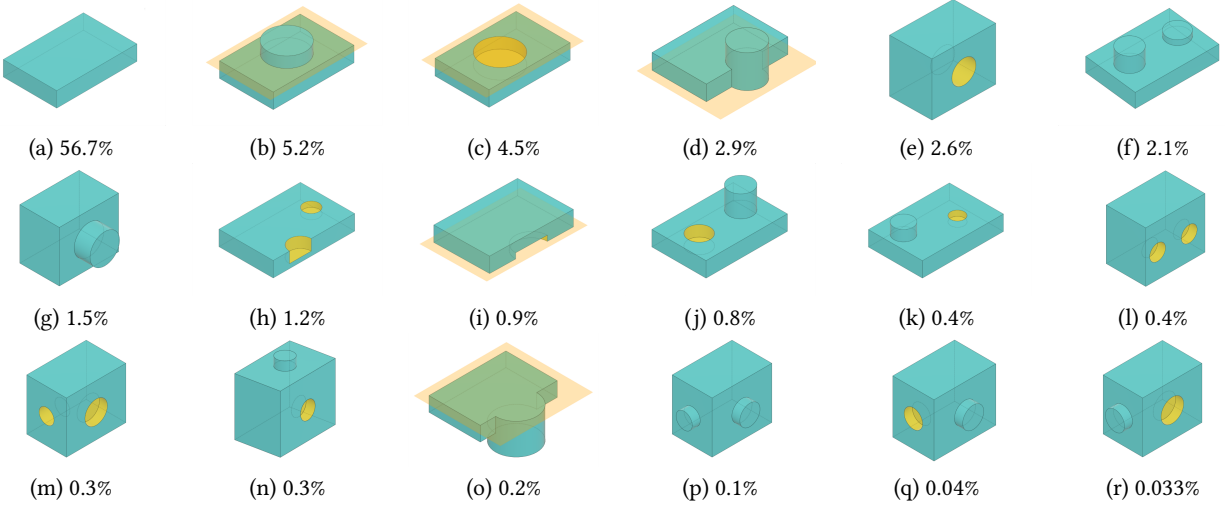


Fig. 5. The frequency of common combinations of extrusions in the DeepCAD dataset. Faces resulting from Boolean subtractions are shown in yellow. Planes shown in orange are constrained to be coincident for pairs of extrusions as described in Section A.3. (a) Single extrusion. (b) Two joined extrusions sharing a common plane. (c) A base extrusion cut by a second extrusion. The end plane of both extrusions is shared. (d) Two joined extrusions sharing a common start plane. (e) An extrusion with an orthogonal cut. (f–r) Other combinations of extrusions.

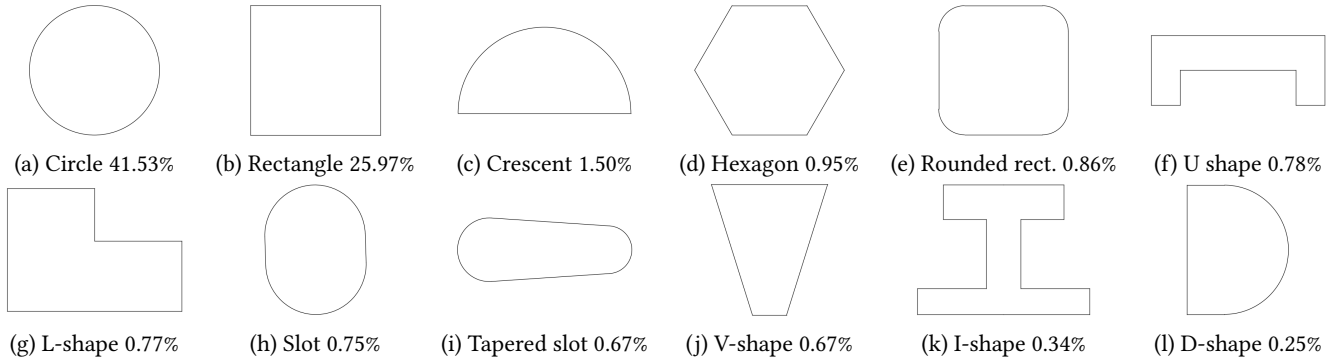


Fig. 6. Examples of the constrained parametric sketches used to generate profile loops in this work. The graph hashing procedure described in section A.2 was used to find profile loops in the DeepCAD dataset which have the same sequence of lines, arcs and important geometric properties (axis alignment, tangent continuity). The percentage of all DeepCAD profile loops with the same graph hash is shown.

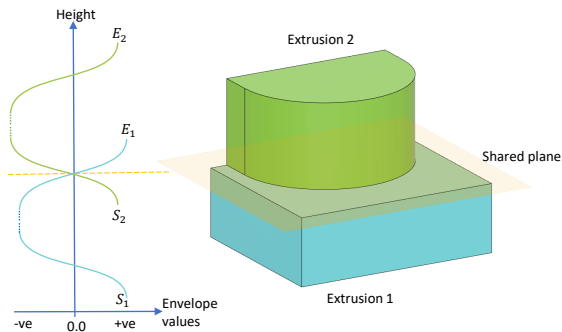


Fig. 7. The envelope array values for the two extrusions sharing a common plane in the same configuration as Figure 5b.

3D autoencoder was trained on 3 Quadro RTX 6000s. The minimum in the validation loss was achieved after 12h. Inference requires 2.2s per solid on CPU. The majority of this time is spent in the profile fitting step which could be further optimized using a constraint solver mode which reuses some parts of the initial processing of the constraints to make subsequent solves faster.

A.10 Failure cases

In this section we discuss the failure modes for our approach. There are three common failure modes which we observe in the generated results. Figure 8a shows a failure mode where the network correctly decodes the profile image, however the search, retrieval and fitting procedure results in profile loops which touch or intersect. This causes the downstream CAD construction to fail the validity

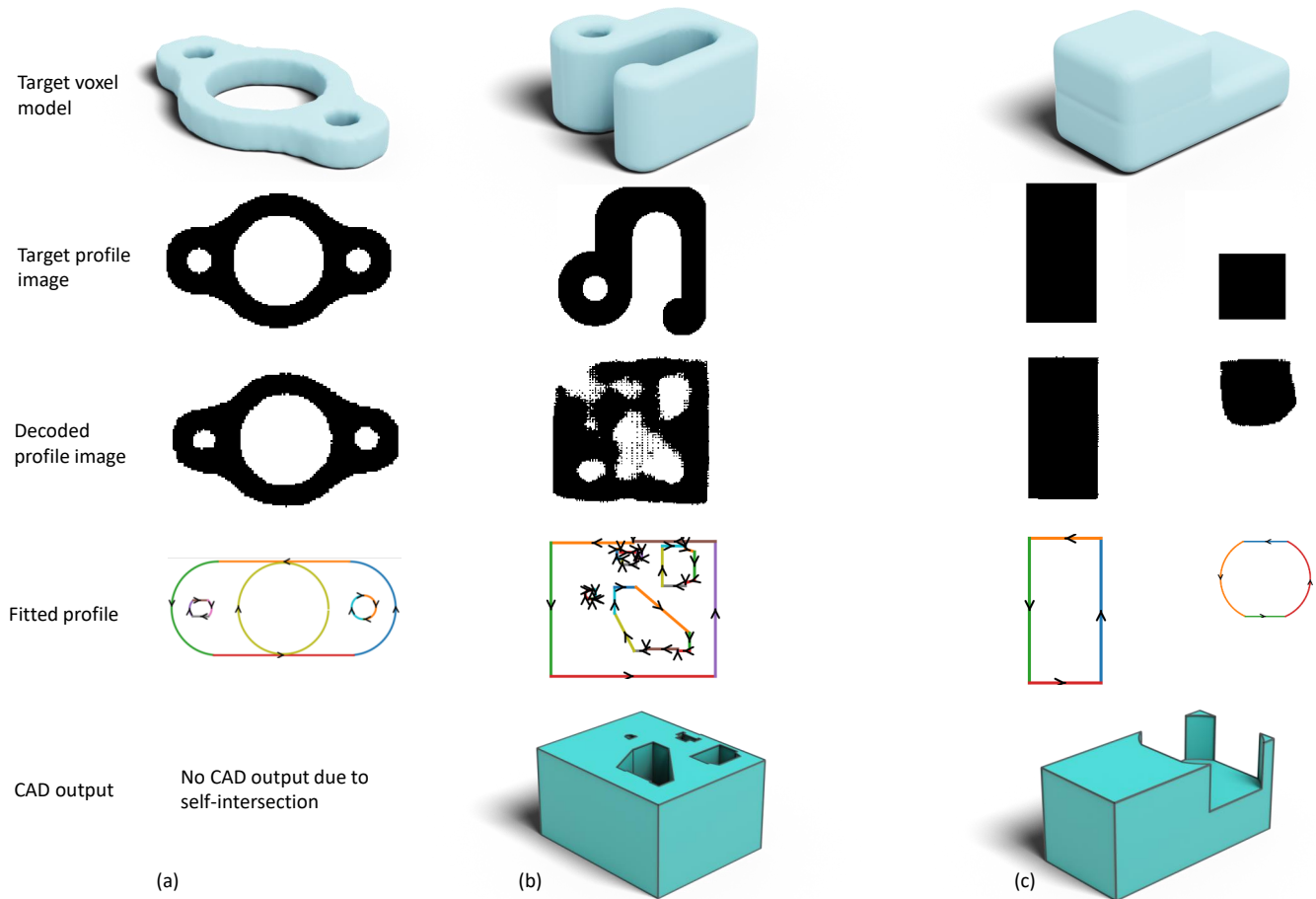


Fig. 8. a) Failure mode in which the retrieved profiles self-intersect causing the constructed CAD to be invalid. b) Failure mode in which the network is unable to decode the correct profile. c) Failure mode in which misalignment between profiles causes the final solid model to be visually very different from the target. This usually happens when one extrusion is subtracted from another.

checking and valid CAD output cannot be generated. Further processing of the 2D curves could detect cases like this and remove intersections, however touching loops as shown in this example would result in a non-manifold B-Rep body if extruded.

Figure 8b shows a case where the network fails to decode the profile with sufficient fidelity and the search, retrieval and fitting procedure produces loops which do not faithfully represent the target shape. This typically happens for complex profiles. We have observed that this failure mode is more common when training on a subset of the available data and believe training on larger collections of sketch and extrude data would alleviate this problem.

The third common failure mode is shown in 8c. Here multiple extrusions are combined to build the CAD model and having the correct alignment between the extruded profile is important for the quality of the result. In this example the shape has almost equal probability of being made as the union of two extrusions (5.2% of DeepCAD files) or by subtracting one extrusion from the other (4.5% of DeepCAD files). The ground truth for this specific model was generated as the union of two extrusions, while the decoder module

giving the result which most closely approximates the input voxels attempts to create the shape by subtracting a small profile from the larger one. For the subtraction to remove material as expected, it is important the second profile is not smaller than the target shape. In this case the fidelity of the second decoded profile is insufficient for the fitted curves to align with the main profile, resulting in incorrect spikes at the far end of the regenerated CAD. While the union and subtraction construction sequences are equally valid for designers, automated algorithms may derive some benefit from building models as the union of extrusions where possible. It is hoped that this insight is useful for researchers attempting to derive canonical CAD sequences from real world data.

A.11 2D profile interpolation

The search, retrieval and fitting algorithm can also be used to interpolate between different CAD profiles. Given a start and end profile, these can be embedded using the CNN encoder described in Appendix A.5. The embeddings of the profiles, z_{start} and z_{end} can

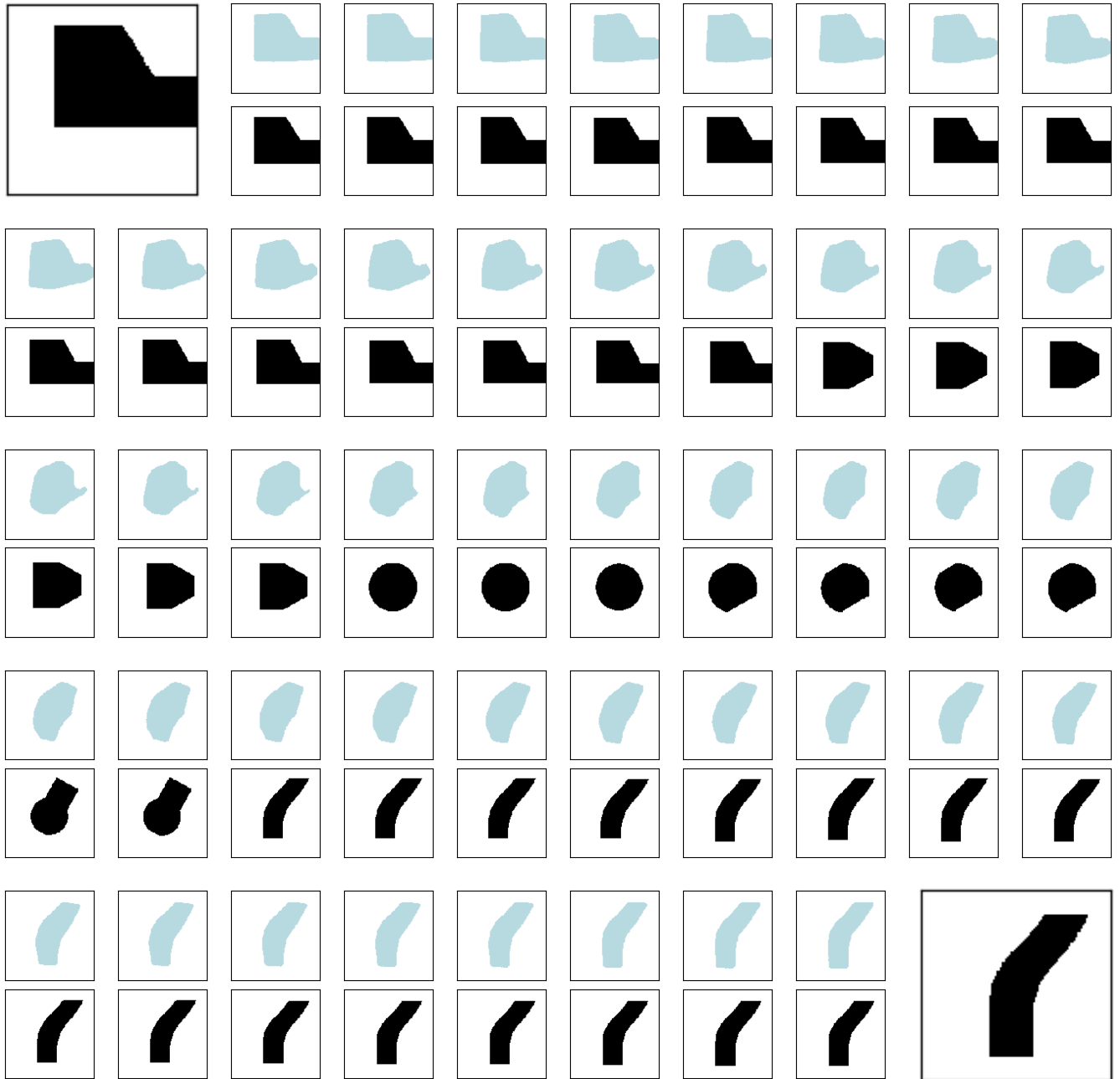


Fig. 9. 2D profile interpolation. The shapes of the start and end profiles are show in the top left and bottom right hand sides. The light blue shapes show the output of the 2D deconvolution decoder and the black shapes are the retrieved and fitted profiles. Changing the sketch parameters in this way allows the geometry to evolve smoothly, demonstrating the advantage of having a parametric model. The gradual changes in the sketch geometry is better shown in the video in the supplementary material.

then be linearly interpolated

$$z(t) = (1 - t) z_{start} + t z_{end} \quad (7)$$

where the parameter t is in the range $[0, 1]$. For any intermediate value of $z(t)$, an approximate shape can then be decoded using the

deconvolution decoder described in Appendix A.6. These decoded shapes are shown in light blue in Figure 9. While they produce a smooth interpolation between the start and end shapes, the geometry becomes rounded and distorted and the intermediate shapes do not look like geometry which a CAD designer would create.

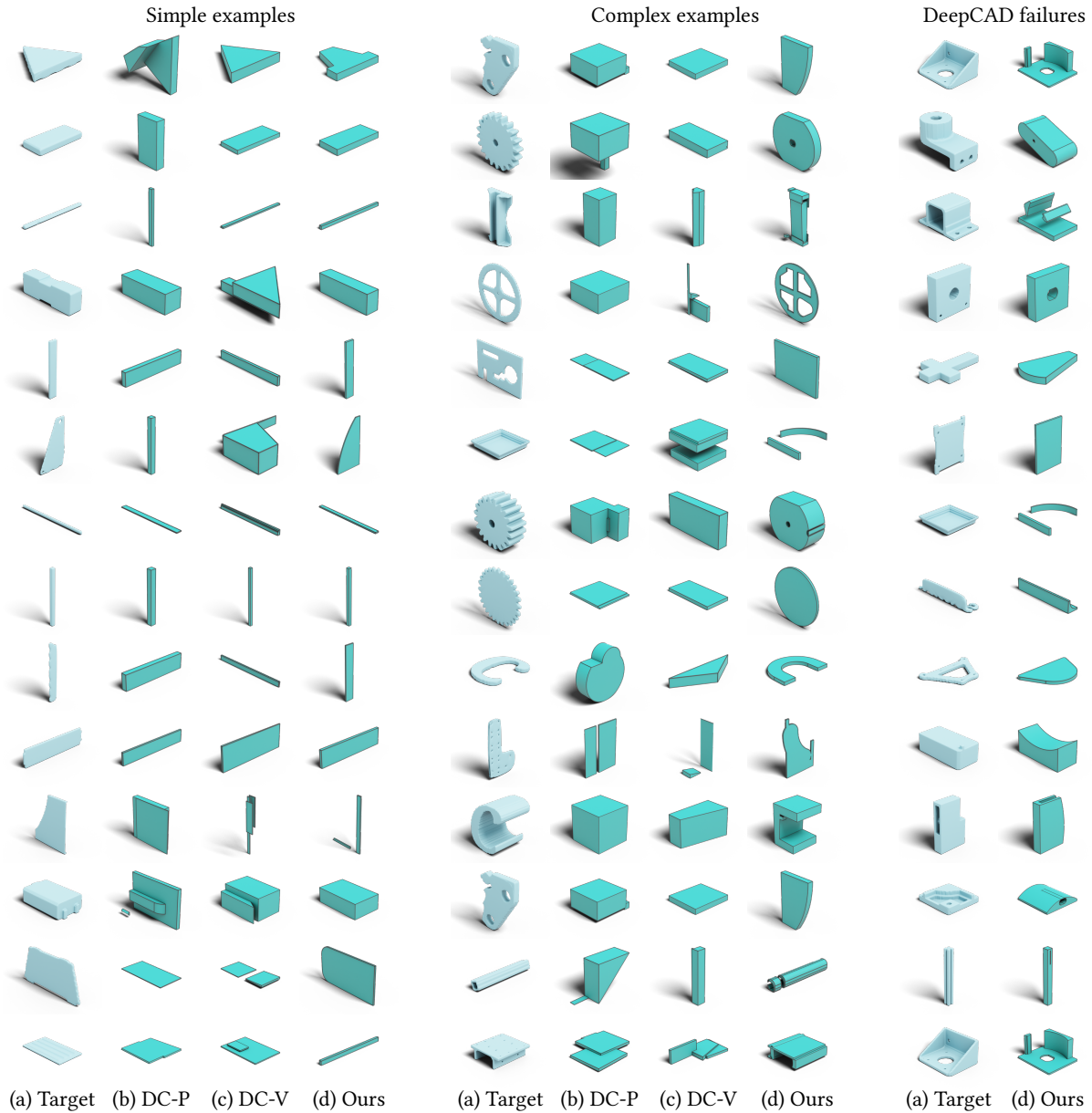


Fig. 10. Reconstructed models using voxelized solids from the ABC dataset as the target geometry. The solids were sampled at random from those created during the experiment described in Section 6.2. On the left we show models which all methods could create for simple target shapes. In the middle we show more complex examples. On the right we show examples where DeepCAD failed to generate valid solids, but our method was able to do so successfully. (a) Target voxel model. (b) DeepCAD using the point cloud encoder. (c) DeepCAD using the voxel encoder. (d) Ours

To recover CAD-like shapes we first create an embedding for each variation of the sketch templates described in Appendix A.2. This gives a set of embeddings $Z = \{z_0, z_1, \dots, z_{1689}\}$. We then choose the template variation such that $|z_i - z(t)|$ is the minimum for any $z_i \in Z$. This variation of the template includes the sketch geometry, topology and also the sketch parameters which gave rise to the specific variation of the geometry with embedding z_i . We then use the simplex algorithm [Nelder and Mead 1965] to fine tune

the parameters of the sketch so that the extracted profile has the highest IoU with the approximate shape decoded from $z(t)$. This results in the CAD profiles shown in black in Figure 9. In regions where the same sketch is retrieved for consecutive frames, this procedure allows the parameters of the sketch to be varied smoothly, providing a very gradual distortion of the retrieved and fitted shape. This behavior is best observed in the video in the supplementary

material, as the gradual changes in parameters are difficult to show in static images.

An important property of this procedure is that the retrieved and fitted shapes always have the appearance of a profile which a CAD designer would create. The constraints inside the sketches maintain certain important aspects of the design intent like horizontal and vertical lines and tangent continuity. While the algorithm attempts

modify the sketch parameters to give rise to the smoothest interpolation possible, there are still sudden changes to the profile when the closest sketch template changes. Collections of constrained sketches which minimize the size of these jumps in the shape during interpolation are desirable as they can be thought of as providing a better *covering* of the space of possible target shapes.