Automatic Reverse Engineering: Creating computer-aided design (CAD) models from multi-view images

Henrik Jobczyk and Hanno Homann

Hannover University of Applied Sciences, Germany
henrik.jobczyk@stud.hs-hannover.de, hanno.homann@hs-hannover.de

Abstract. Generation of computer-aided design (CAD) models from multi-view images may be useful in many practical applications. To date this problem is usually solved with an intermediate point-cloud reconstruction and involves manual work to create the final CAD models. In this contribution, we present a novel network for an automated reverse engineering task. Our network architecture combines three distinct stages: A convolutional neural network as the encoder stage, a multi-view pooling stage and a transformer-based CAD sequence generator.
The model is trained and evaluated on a large number of simulated input images and extensive optimization of model architectures and hyperparameters is performed. A proof-of-concept is demonstrated by successfully reconstructing a number of valid CAD models from simulated test image data. Various accuracy metrics are calculated and compared to a state-of-the-art point-based network.
Finally, a real world test is conducted supplying the network with actual photographs of two three-dimensional test objects. It is shown that some of the capabilities of our network can be transferred to this domain, even though the training exclusively incorporates purely synthetic training data. However to date, the feasible model complexity is still limited to basic shapes.

Keywords: computer-aided design (CAD) · multi-view reconstruction · encoder-decoder network.

1 Introduction

Ever since the invention of 3D-printing in the middle of the 20th century, it stimulates the imagination of laypersons and engineers alike. Nowadays this technology is an integral part of the product development cycle in many industries and its application often goes beyond the production of mere prototypes.

Even though online 3D printing services increase availability at affordable prices, their use in everyday life is not straightforward. This work is focuses on the central problem of 3D-printing: The generation of digital 3D objects is a skill requiring specialized technical expertise and training, posing a significant barrier for consumer adoption.

To give a practical example, a simple mechanical part within a bigger and more expensive appliance such as a washing machine or dryer fails and renders the device unusable. The point of failure is identified but the manufacturer can not offer a spare part. If the user could simply take a few photos using a smartphone camera and have a computer-aided design (CAD) model created automatically by software, the problem could be solved in a short time at minimal financial and environmental cost.

This work proposes an end-to-end solution for this reverse engineering problem, which is to our knowledge the first of its kind. Our network architecture is illustrated in Figure 1 and will be described in detail further below after revisiting the state-of-the-art. For proof-of-concept, our model was trained on a large number of renderings from simulated CAD objects. Our results indicate that the image-based approach may outperform a current point-based method. Finally, two real world objects were photographed and reconstructed.

Our main contributions are: (1) We present the first end-to-end model to generate CAD sequences from multi-view images, (2) comparison of two different multi-view fusion strategies, and (3) initial results on real-world photos.

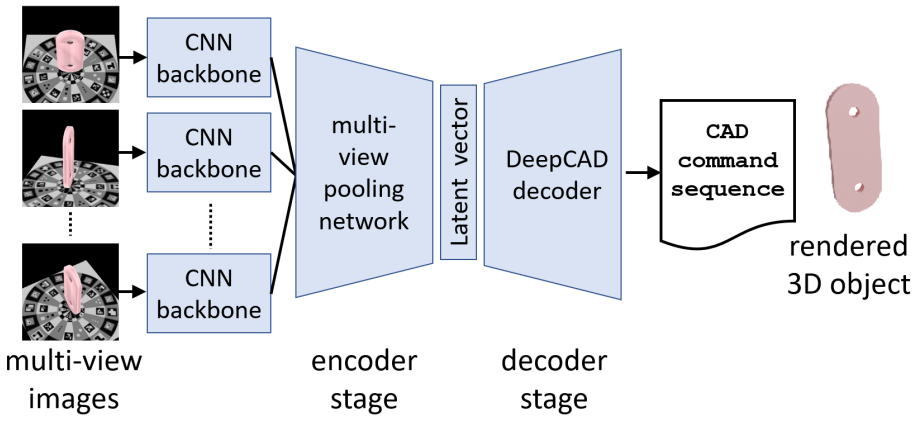

Fig. 1. ARE-Net architecture: Input images taken from multiple view angles are fed into an encoder-decoder network to generate CAD sequence file. Multi-view fusion is facilitated (a) using a fully-connected network (FCN) or (b) using a gated recurrent unit (GRU) to allow varying numbers of input images. The decoder part of the DeepCAD auto-encoder is employed as the generative decoder.

2 Related work

2.1 Traditional photogrammetry approaches to reconstructing CAD models

Photogrammetry is frequently deployed as an image-based technique to measure three-dimensional shapes using inexpensive cameras. The most common monocular approaches are based on the Structure from Motion (SfM) method first described in [35]. Here, the software is provided with several images from different perspectives and then computes a point-cloud of the object of interest.

Automatically extracting a CAD model from a point-cloud is however not straight-forward. For example, the professional AutoCAD software can import but not post-process point clouds as of today [3]. Thus far, CAD model creation mostly remains a manual task.

Kim et al. [15] proposed 3D registration of given CAD model using the iterative closest point (ICP) method. Budroni et al. [5] have demonstrated the fitting of planar surfaces to point clouds for reconstructing of 3D models of interior rooms. More recently, Lui [19] proposed automatic reverse-engineering of CAD models from points clouds by iteratively fitting primitive models based on the RANSAC algorithm. In conclusion, there are few existing approaches which are however domain-specific. Instead, a neural-network based approach might generalize better in the long term.

2.2 Learning-based object reconstruction

Detection of 3D objects from multiple view perspectives has been addressed by Rukhovich et al. [33]. Similar to [39], they used a fully convolutional network. Notably the number of monocular images in their multi-view input can vary from inference to inference, offering high versatility. This is achieved by extracting features with conventional a Convolutional Neural Network (CNN), followed by pooling and back-projecting into a 3D volumetric space. In this space, bounding boxes are predicted by three-dimensional convolutions.

For 3D surface reconstruction, deep learning models have been suggested for different kinds of object representations, including point clouds [1,12,8,48,49,6,21], triangle meshes [38,10,23,26], voxel grids [18,42,47], cubic blocks [46], parametric surfaces [34,40,13,16,17,45], and signed distance fields (SDFs) [28,14]. The majority of the studies above (e.g. [1,26,28,14]) use auto-encoders, with a feature bottleneck between an encoder and a decoder stage. This network architecture also allows to simplify training by separating the two stages. To date, discrete CAD models have not been investigated for 3D surface representation.

2.3 Multi-view convolutional networks

A 3D multi-view CNN (MVCNN) for object classification was introduced by Su et al. [36]. They provide a certain number of images taken of the object as input to a common CNN and pool the extracted features using an element-wise

maximum operation. The pooled information is then processed by a second CNN and a final prediction is made. Notably they conclude that inputting 12 evenly spaced perspectives offers the best trade-off between prediction accuracy and memory as well as time resources.

Their concept as been studied for classifying 3D shapes from point clouds [22,31]. In general, working with MVCNNs seems to be a viable approach for extracting information from 3D scenes. Leal et al. [37] compared different 3D shape classifiers, identifying MVCNNs as superior to other methods due to better generalizability and outperforming several point-based and voxel-based approaches. Consequently, this approach will be followed in this work.

2.4 Recurrent convolutional networks

While MVCNNs showed good results for classification tasks, the simple pooling methods (e.g. element-wise max-pooling [36]) might allow a single view to overrule all other views. Geometric information not visible in some images might be lost for a 3D reconstruction task. Hence, we alternatively consider Recurrent CNNs as a more preservative information extractor.

Zreik et al. [50] used a Recurrent Neural Network (RNN) for spacial aggregation of extracted information from 3D angiography images after pre-processing by a 3D-CNN. Liu et al. [20] combined a traditional 2D CNN backbone and an RNN to synthesize multi-view features for a prediction of plant classes and conditions. After extensive experiments, they conclude that a combination of MobileNet as a backbone and a Gated Recurrent Unit (GRU) delivers the best trade-off of classification accuracy and computational overhead. Hence, GRUs will be evaluated in this study for multi-view pooling.

2.5 Generation of CAD representations

Even though most methods described above generate three-dimensional data, none of them directly attempts CAD file generation by means of generating a construction sequence comparable to manual CAD design. Thus their resulting models cannot easily be modified by an average user. However, recent research has started to address the direct generation of parametric 2D CAD models:

Willis et al. [41] first proposed generative models for CAD sketches, producing curve primitives and explicitly considering topology. SketchGen [27] generates CAD sketches in a graph representation, with nodes representing shape primitives and edges embodying the constraints. Similarly, Ganin et al. [9] utilized off-the-shelf data serialization protocols to embed construction sequences parsed from the online CAD editor Onshape [24].

DeepCAD by Wu et al. [44] was the first approach going beyond the 2D domain of CAD sketch generation. They formulated CAD modeling as a generation of command sequences, specifically tailored as an input to a transformer-based auto-encoder. The publicly available Onshape API was used to build a large dataset of 3D object models for training.

Each object is represented by a CAD sequence, consisting of three common types of commands: (1) Creation of a closed curve profile ("sketch") on a 2D plane, (2) 3D extrusions of such sketches and (3) boolean operations between the resulting 3D objects. Each of the CAD commands supports a number of parameters, which may be a mixture of continuous and discrete values. To conform with their neural network, Wu et al. sort each command's parameters into a generalized parameter vector and all continuous parameters are quantized to 8-bits. The maximum number of commands in a given CAD construction sequence was limited to 60, corresponding to the longest sequence length in the dataset.

These CAD sequences are processed by an auto-encoder, trained to compress a given CAD model into a latent vector (dimension of 256) and then to reconstruct the original model from that embedding. This means, a random but valid CAD object can be constructed using a given 256-dimensional latent vector. In this work, chose the decoder part of DeepCAD as the generative stage of our new model as introduced next.

3 Methods

3.1 Network architecture

We introduce a novel network architecture for end-to-end generation of CAD models from multiple input images. The network is composed of three stages: (1) a CNN encoder backbone to extract information from each input image individually, (2) a pooling network that aggregates this information into a common latent vector, and (3) a generative decoder network constructing the output CAD sequences. This network structure is illustrated in Figure 1.

Considering its successful track record in object detection and classification as well as its small size, we chose the residual network architecture (ResNet) [11] as our encoder backbone. As the visual complexity of our input images is relatively low, we assumed that a smaller, more shallow variant of the network should suffice. Thus only its smallest two variants were evaluated, namely ResNet-18 and ResNet-34. The input image size is adjustable by means of ResNet's adaptive average pooling layer. In this work, we used 128x128 monochrome as well as 224x224 RGB input images. The output of the last fully connected layer, a vector of fixed length 512, is fed into the pooling network. All input views are processed by the backbone network individually but share the same parameters.

The task of the multi-view pooling stage is to combine the information from multiple views. We evaluated two different network architectures during the experiments: (a) a simple feed-forward fully connected network (FCN) as a baseline model and (b) a gated recurrent unit (GRU). Following [7] and [20], we assume that a recurrent pooling approach should perform favorable, even though its training is inherently more challenging [29] because of the possible vanishing and exploding gradient problems.

The FCN pooling network concatenates the outputs of all backbone CNNs and propagates them through a numbers of layers (1 to 6 layers were evaluated)

of linearly decreasing size with a final layer size of 256. This forms the latent vector compatible to the subsequent DeepCAD decoder network.

Unlike the FCN pooling which processes all input views simultaneously, the alternative GRU pooling receives the input views from the backbone CNN sequentially one after the other. This makes it more suitable for varying numbers of images. For evaluation of the GRU pooling stage, we tested different numbers of layers (1 to 8) of identical dimension, different temporal pooling strategies (mean, max, last) and different layer dimensions (64, 128, 256, 512, 1024, 2048). A single fully connected layer is used to achieve the latent vector size of 256.

Both pooling network variants use rectified linear units (ReLU) as their non-linear activation function in all layers except the last. The final layer generates the latent vector. Here the hyperbolic tanget function (*tanh*) is utilized as it provides output in the range $[-1, 1]$ as required for the DeepCAD decoder network.

The final stage of the ARE-Net is formed by the decoder from the DeepCAD library [43] which generates CAD construction sequences from the 256-dimensional latent vector.

3.2 Two-stage training

Training was performed in two stages: First, the full DeepCAD auto-encoder was pre-trained as described in [44]. After this training, the final latent vector of each CAD object from the training set was saved. Second, simulated image views were rendered from the ground truth CAD sequences and used to train our backbone and multi-view pooling networks. As the loss function, we used the mean-squared error between the predicted latent vectors of the simulated images and the ground-truth latent vectors from the first training stage. We employed the ADAM-optimizer, using 10 epochs during hyper-parameter optimization and 140 epochs for the final model.

4 Experimental setup

4.1 Training data

Training images were generated from the DeepCAD dataset consisting of 178,238 CAD models. From each CAD sequence, a 3D mesh object and two different projection datasets were generated: (1) A simple dataset of 128x128 grayscale images from 10 fixed and evenly spaced view angles as shown in Figure 2. (2) A complex dataset of 256x256 RGB images with random but uniform object color from 24 randomly spaced viewing angles. In the second dataset the photogrammetry ground-plane from [4] was used as a base on which each model rests. It is composed of non-repeating patterns and is used as a turntable for real objects during the final real world test. The intention is to provide the model with additional information on orientation and scale of the objects, otherwise lost due to the random viewing angles.

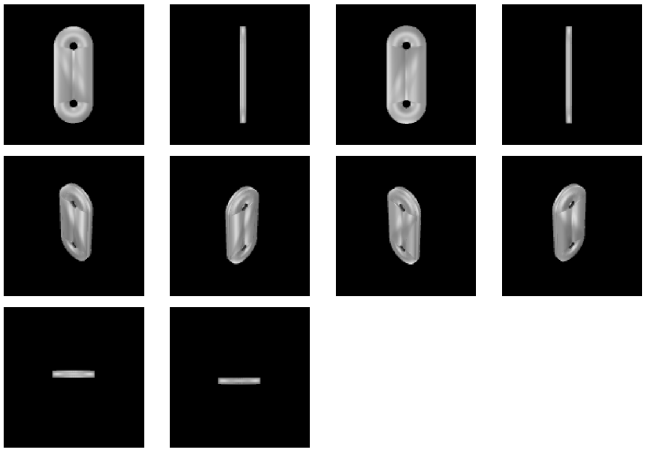

Fig. 2. Example of training images from one CAD model: (top row) central view from four sides, (middle row) elevated view, (bottom row) top and bottom views.

For training on the simple dataset, all 10 images were used. When training on the complex dataset, a random selection of 5 to 20 images was chosen. To allow an unbiased comparison of our network to former work by the DeepCAD researchers, the same training-, validation- and testing-split (90%-5%-5%) used in [44] was applied.

4.2 Hyper-parameter Optimization

Our model contains several hyper-parameters requiring optimization. General parameters are the learning rate, drop out ratio, weight decay and the number of ResNet backbone layers. The parameters of the two pooling networks are the number and dimensions of layers. For the GRU network, the temporal pooling strategy (mean, max, last) also needed investigation. In order to identify suitable hyper-parameters such as network attributes and training parameters which remain constant during any given training run an incremental experimentation procedure is followed. For hyper-parameter optimization, the Optuna library [25] was used. It allows for efficient search through the high dimensional search space and automatically records results and useful statistics.

4.3 Accuracy metrics

To compare the accuracy of the predicted CAD models, three different metrics were employed: The command accuracy ACC_{cmd} measures the agreement of the predicted CAD command type $\hat{t}_i$ with the ground truth command type t_i for a CAD construction sequence of N_c steps:

$$ACC_{cmd} = \frac{1}{N_c} \sum_{i=1}^{N_c} \left(t_i == \hat{t}_i\right) \tag{1}$$

While ACC_{cmd} measures that fraction of correct commands, the correctness of the continuous parameters of each command shall also be evaluated. The parameter accuracy ACC_{param} quantifies the agreement of a predicted 8-bit CAD parameter $\hat{p}_{i,j}$ to its ground-truth counterpart $p_{i,j}$. Only correctly predicted commands $N_{c2} \leq N_c$ were evaluated and a threshold of $\eta = 3$ was used, as suggested in [44]:

$$ACC_{param} = \frac{1}{N_{c2}} \sum_{i=1}^{N_{c2}} \sum_{j=1}^{|\hat{p}_i|} \left(|p_{i,j} - \hat{p}_{i,j}| \leq \eta\right) \tag{2}$$

For geometric evaluation of the 3D model, the so-called Chamfer Distance CD was used [30,2]. It computes the shortest distance of one point x on the surface S_1 of the predicted object to the closest point y on the surface S_2 of the ground-truth object. This is carried out in both directions. In this work, 2000 surface points were evaluated per model.

$$CD = \frac{1}{S_1} \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \frac{1}{S_2} \sum_{x \in S_2} \min_{y \in S_1} ||y - x||_2^2 \tag{3}$$

4.4 Benchmark comparison

As no other method generating CAD models is known to us, comparison is performed using the following two methods: (1) The original DeepCAD auto-encoder is fed with ground-truth CAD-sequences to encode a latent vector and decoded again. The predicted CAD sequence is then evaluated by the accuracy metrics described above. By using loss-less input CAD sequences, this approach represents the ideally achievable results in our comparison and will be referred to as the "baseline".

(2) For a more realistic comparison, the PointNet++ encoder [32] was evaluated as a state-of-the-art method. Point-clouds were sampled from the ground-truth 3D objects. The PointNet++ encoder was used to map the point-clouds into a latent vector and then processed by the Deep-CAD decoder as proposed by [44].

4.5 Reconstruction from photographic images

For an initial assessment of the performance of our method on real world images, two test objects were chosen: a cardboard box representing a very simple case and a camera mount as a more complex example. Both are intentionally of uniform color to match the simulated objects seen during training. The objects were placed on a paper version of the photogrammetry ground plane. Then 20 pictures from varying perspectives were taken by a smartphone camera while changing

the inclination angle relative to the ground plane and rotating a turntable underneath the object. The image background behind the objects was then cropped away manually. All pictures were sized down to 224x224 pixels and passed into the Automatic Reverse Engineering Network (ARE-Net) with GRU pooling as trained on the simulated complex dataset.

5 Results

The best performing hyper-parameters are summarized in Table 1. On the simple dataset the GRU with a shallow ResNet18 backbone had sufficient distinguishing power, whereas ResNet34 performed better for the simpler FCN network as well as for the GRU for the complex dataset. Three FC layers were optimal for FCN pooling, but more than one layer didn't increase performance of the GRU pooling stages. As for the GRU-specific parameters, sightly larger networks proved favorable for the complex dataset.

Pooling network	FCN	GRU	GRU
Dataset	simple	simple	complex
Learning rate	$1.3 \cdot 10^{-4}$	$4.8 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$
Drop out	4.8%	16.1%	17.2%
Weight decay	$5.45 \cdot 10^{-5}$	$3.18 \cdot 10^{-6}$	$4.38 \cdot 10^{-6}$
Backbone	ResNet34	ResNet18	ResNet34
FC layers	3	1	1
GRU pooling	-	*max*	*last*
GRU layers	-	1	2
GRU dimension	-	1024	2048

Table 1. Best hyper-parameters found by our optimization.

Table 2 compares the accuracy metrics of our models using the optimized hyper-parameters. It stands out that the GRU pooling network trained on the simple dataset achieved the best overall performance. It reaches an ACC_{cmd} of 92.8%, an ACC_{param} of 78.8% and a median CD of $1.75{\cdot}10^3$. However, the fraction of 18.4% of CAD models that could not be constructed is notably worse than for the point cloud encoder. The percentage of invalid CAD topologies is reported as "CAD model invalid". valid. An invalid sequence may occur, for example, if a curve sketch command is not followed by a 3D extrusion. This tends to occur more often for longer command sequences.

The ARE-Net models trained on the simple datasets surpass the one trained on the complex data. The random variation of perspectives and number of input images during training represent a harder problem which did not provide an advantage in this comparison.

The accuracy on the test set of the ARE-Net with GRU pooling is plotted in Figure 3 as a function of the number of input images. Above 13 images the

Method	$ACC_{cmd} \uparrow$	$ACC_{param} \uparrow$	*median CD* $\downarrow$	*CAD model invalid* $\downarrow$
ARE-Net FC (simple data)	92.14%	74.2%	$4.21 \cdot 10^3$	18.1%
ARE-Net GRU (simple data)	**92.83%**	**78.8%**	$\mathbf{1.75} \cdot 10^3$	18.4%
ARE-Net GRU (complex data)	92.78%	74.6%	$4.07 \cdot 10^3$	18.8%
DeepCAD PointNet++	84.95%	74.2%	$10.3 \cdot 10^3$	**12.1%**
Baseline:				
DeepCAD auto-encoder	99.50%	98.0%	$0.75 \cdot 10^3$	2.7%

Table 2. Quantitative results of CAD reconstruction of the presented ARE-Net, DeepCAD with point cloud network and the DeepCAD auto-encoder.

accuracy barely increases, which is in line with [36] describing 12 images as a useful lower bound, beyond which the accuracy of their network levels.

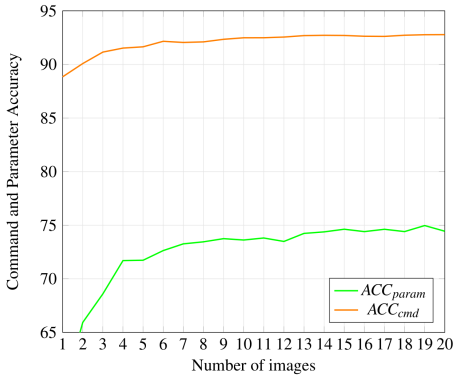

Fig. 3. Accuracy results for different numbers of input images passed into the ARE-Net using the complete object test set.

Figure 4 compares the reconstructed geometries. The following observations can be made: A variety of reconstructions is quite successful. Often times the network seems to "comprehend" the basic form of the shape present, but lacks the ability to exactly reproduce it quantitatively. For example, regarding the yellow object in the bottom right corner of Figure 4, it is clear that the model has recognized the basic shape of the plate and manages to reproduce it quite well. It also extracts the correct number of holes but still fails to reproduce their size and exact position.

Conversely, a fraction of about 18% of more complex ground-truth models could not be successfully reconstructed, some examples are show in Figure 5. Visual comparison shows that these models are generally more complex than their valid counterparts, e.g. containing holes of different diameters or extrusion into different spatial directions.

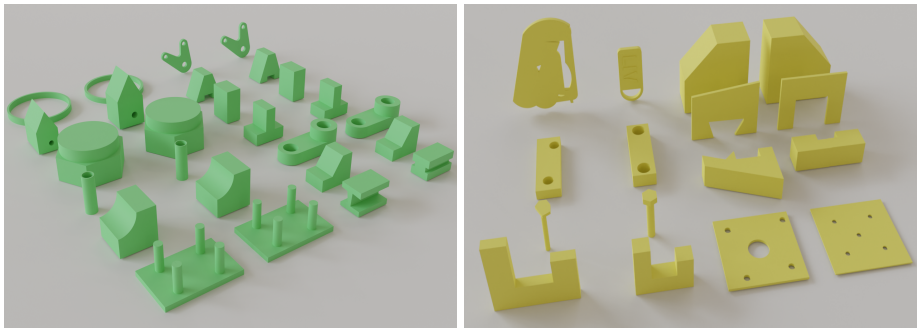

Fig. 4. Random selection from the test set of representative good (green) and poor (yellow) reconstruction results. The model predictions are shown on the left, next to their corresponding ground-truth models.

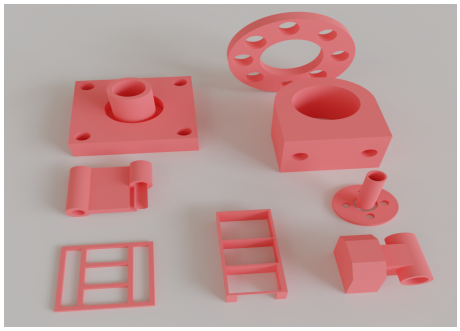

Fig. 5. Random selection from the test set of ground-truth models that could not be successfully reconstructed.

Two representative photos of our real world objects and their reconstructions are shown in Figure 6. The reconstructed CAD sequence of the cardbox is a perfect cube with equal side lengths, up to the 8-bit precision. As for the more complicated camera mount, a valid CAD model could be created from the photos. However, only the basic L-shape is represented by the model. The relative dimensions are inaccurate and details like the screw holes are completely missing. Moreover, the reconstruction exhibits a prominent elongated bar at the bottom which is not at all present in the original model. This second real-world reconstruction was hence only partially successful.

6 Discussion and conclusions

We developed a novel method for end-to-end generation of CAD sequences directly from photographic images using an encoder-decoder network architecture. Models were trained in a two-stage approach on 2D renderings of simulated CAD

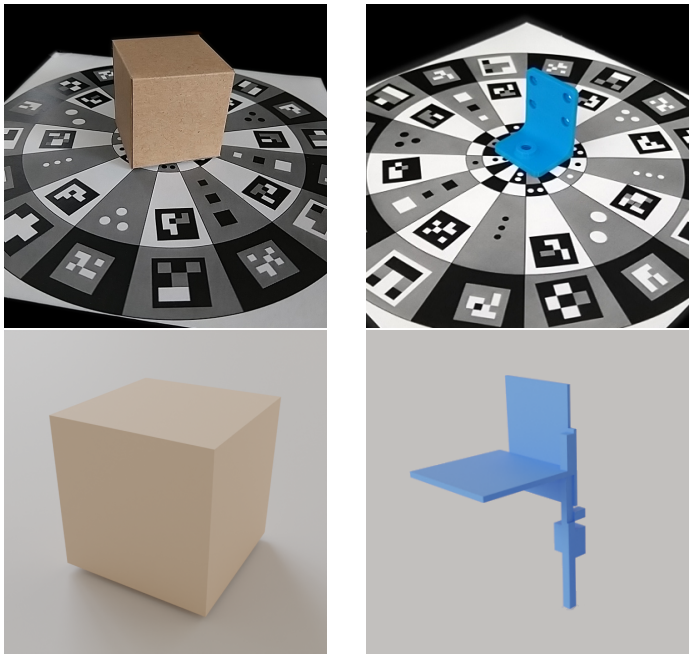

Fig. 6. Real object reconstruction attempts: The top row show selected photos of the two objects placed on the photogrammetry ground-plane (left: cardboard box, right: camera mount angle). The bottom row shows the respective CAD reconstructions.

objects and positively evaluated. A first proof-of-concept of the method on real photos was realized.

Two different multi-view pooling stages were compared: a feed-forward fully-connected network (FCN) and a gated recurrent unit (GRU). A number of hyper-parameters were extensively optimized. Our results show that the additional complexity introduced by the GRU pays off by producing a significant improvement in all three accuracy metrics. Moreover, the GRU takes in the individual images one after the other such that the number of input images can be handled more flexibly. Our experiments confirm the earlier finding [36] that around 12 different views of an object can be considered a practical lower bound, with little improvement above that number.

Comparing our CAD models reconstructed from rendered images of the test set to reconstructions from 3D point-clouds by the state-of-the art PointNet++ encoder, our encoders successfully created valid CAD sequences in more than 80% of the cases which is lower than the success rate of the point-cloud encoder. Regarding the accuracy measures, our encoders outperformed the point-cloud encoder by a large margin.

Most importantly, our work establishes the basic feasibility of image-based reverse engineering of 3D CAD models by neural networks. In future applications

this might reduce the amount of time-consuming work of highly trained engineers or enable untrained laymen to work with CAD technologies for 3D printing previously inaccessible without specialized training.

Current limitations of the approach include that the length of CAD sequences is still limited to 60 commands, hence only supporting relatively simple objects. Also our representation is limited to planar and cylindrical surfaces, while many real-world objects may include more flexible triangle meshes or spline representations.

Furthermore, the exact position and size of object details - especially small holes - must be improved for practical applications. The loss function used to train the DeepCAD decoder network penalizes deviations of the CAD parameters but does not contain a distance metric [44]. We believe that an end-to-end training of the complete model may improve these results, allowing for more specialized loss functions to get a more direct handle on the quantitative sequence parameters.

Future work should also focus on improving the image rendering of the training data. This may include physics-based rendering techniques such as ray-tracing to better simulate real-world cases and the incorporation of reflections, image blur and noise to better mimic an actual picture taken by the end-user. Data augmentation by different backgrounds and model textures should also be considered. Just like the camera view angles, the distance and translation of the object should also be varied. A fine-tuning of the model parameters training with a (limited) set of real-world photos of 3D-printed objects from given CAD models could also be pursued. Finally different backbone and/or pooling architectures, such as attention based techniques could be explored going forward.

Generally the direction proposed in this work seems promising. It will be interesting to see what this or similar approaches will lead to down the line. One may predict that experts and consumers might soon be using parametric, CAD generating 3D-scanning-applications, just as naturally as optical character recognition (OCR) is used today, saving countless hours of repetitive work and providing unpreceded possibilities of interaction and creation in this three-dimensional world.

Acknowledgements

We would like to thank Rundi Wu and his co-workers for openly sharing their ground-braking DeepCAD work and providing extensive support materials such as their dataset, the generative CAD decoder, the point-cloud encoder and evaluation metrics.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: International conference on machine learning. pp. 40–49. PMLR (2018)

2. Agarwal, N., Yoon, S.e., Gopi, M.: Learning Embedding of 3D models with Quadric Loss (Jul 2019), http://arxiv.org/abs/1907.10250, number: arXiv:1907.10250 arXiv:1907.10250 [cs]
3. Autodesk-Support: Which point cloud file formats can autocad import? (2022), https://www.autodesk.com/support/technical/article/caas/sfdcarticles/sfdcarticles/Which-point-cloud-file-formats-can-AutoCAD-import.html Access Date: 2023-07-18
4. Boessenecker, R.: The Coastal Paleontologist, atlantic edition: Photogrammetry turntable backgrounds - free to use (Jun 2020), https://coastalpaleo.blogspot.com/2020/06/photogrammetry-turntable-backgrounds.html, uRL: https://coastalpaleo.blogspot.com/2020/06/photogrammetry-turntable-backgrounds.html Access Date: 2022-04-25
5. Budroni, A., Boehm, J.: Automated 3d reconstruction of interiors from point clouds. International Journal of Architectural Computing **8**(1), 55–73 (2010)
6. Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N., Hariharan, B.: Learning Gradient Fields for Shape Generation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020, vol. 12348, pp. 364–381. Springer International Publishing, Cham (2020), https://link.springer.com/10.1007/978-3-030-58580-8_22, series Title: Lecture Notes in Computer Science
7. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/D14-1179, http://aclweb.org/anthology/D14-1179
8. Fan, H., Su, H., Guibas, L.: A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2463–2471. IEEE, Honolulu, HI (Jul 2017). https://doi.org/10.1109/CVPR.2017.264, http://ieeexplore.ieee.org/document/8099747/
9. Ganin, Y., Bartunov, S., Li, Y., Keller, E., Saliceti, S.: Computer-Aided Design as Language. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 5885–5897. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper/2021/file/2e92962c0b6996add9517e4242ea9bdc-Paper.pdf
10. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A Papier-Mache Approach to Learning 3D Surface Generation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 216–224. IEEE, Salt Lake City, UT, USA (Jun 2018). https://doi.org/10.1109/CVPR.2018.00030, https://ieeexplore.ieee.org/document/8578128/
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778. IEEE, Las Vegas, NV, USA (Jun 2016). https://doi.org/10.1109/CVPR.2016.90, http://ieeexplore.ieee.org/document/7780459/
12. Insafutdinov, E., Dosovitskiy, A.: Unsupervised learning of shape and pose with differentiable point clouds. Advances in neural information processing systems **31** (2018)
13. Jayaraman, P.K., Sanghi, A., Lambourne, J.G., Willis, K.D.D., Davies, T., Shayani, H., Morris, N.: UV-Net: Learning from Boundary Representa-

tions (Apr 2021), http://arxiv.org/abs/2006.10211, number: arXiv:2006.10211 arXiv:2006.10211 [cs]
14. Jiang, Y., Ji, D., Han, Z., Zwicker, M.: Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1251–1261 (2020)
15. Kim, C., Lee, J., Cho, M.: Fully automated registration of 3d cad model with point cloud from construction site. In: Proceedings of the 28th International Symposium on Automation and Robotics in Construction, ISARC 2011. pp. 917–922 (2011)
16. Lambourne, J.G., Willis, K.D.D., Jayaraman, P.K., Sanghi, A., Meltzer, P., Shayani, H.: BRepNet: A topological message passing system for solid models. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12768–12777. IEEE, Nashville, TN, USA (Jun 2021). https://doi.org/10.1109/CVPR46437.2021.01258, https://ieeexplore.ieee.org/document/9578870/
17. Li, C., Pan, H., Bousseau, A., Mitra, N.J.: Sketch2CAD: sequential CAD modeling by sketching in context. ACM Transactions on Graphics **39**(6), 1–14 (Dec 2020). https://doi.org/10.1145/3414685.3417807, https://dl.acm.org/doi/10.1145/3414685.3417807
18. Liao, Y., Donne, S., Geiger, A.: Deep Marching Cubes: Learning Explicit Surface Representations. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2916–2925. IEEE, Salt Lake City, UT (Jun 2018). https://doi.org/10.1109/CVPR.2018.00308, https://ieeexplore.ieee.org/document/8578406/
19. Liu, J.: An adaptive process of reverse engineering from point clouds to cad models. International Journal of Computer Integrated Manufacturing **33**(9), 840–858 (2020)
20. Liu, X., Xu, F., Sun, Y., Zhang, H., Chen, Z.: Convolutional Recurrent Neural Networks for Observation-Centered Plant Identification. Journal of Electrical and Computer Engineering **2018**, 1–7 (2018). https://doi.org/10.1155/2018/9373210, https://www.hindawi.com/journals/jece/2018/9373210/
21. Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N.J., Guibas, L.J.: StructureNet: hierarchical graph networks for 3D shape generation. ACM Transactions on Graphics **38**(6), 1–19 (Dec 2019). https://doi.org/10.1145/3355089.3356527, https://dl.acm.org/doi/10.1145/3355089.3356527
22. Mohammadi, S.S., Wang, Y., Bue, A.D.: Pointview-GCN: 3D Shape Classification With Multi-View Point Clouds. In: 2021 IEEE International Conference on Image Processing (ICIP). pp. 3103–3107. IEEE, Anchorage, AK, USA (Sep 2021). https://doi.org/10.1109/ICIP42928.2021.9506426, https://ieeexplore.ieee.org/document/9506426/
23. Nash, C., Ganin, Y., Eslami, S.M.A., Battaglia, P.W.: PolyGen: An Autoregressive Generative Model of 3D Meshes (Feb 2020), http://arxiv.org/abs/2002.10880, number: arXiv:2002.10880 arXiv:2002.10880 [cs, stat]
24. Onshape, P.I.: Onshape | Product Development Platform, https://www.onshape.com/en/, uRL: https://www.onshape.com/ Access Date: 2022-04-12
25. Optuna, P.N.I.: Optuna - A hyperparameter optimization framework, https://optuna.org/, uRL: https://optuna.org/ Access Date: 2022-06-24
26. Pan, J., Han, X., Chen, W., Tang, J., Jia, K.: Deep mesh reconstruction from single rgb images via topology modification networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9964–9973 (2019)

27. Para, W., Bhat, S., Guerrero, P., Kelly, T., Mitra, N., Guibas, L.J., Wonka, P.: SketchGen: Generating Constrained CAD Sketches. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 5077–5088. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper/2021/file/28891cb4ab421830acc36b1f5fd6c91e-Paper.pdf
28. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 165–174 (2019)
29. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training Recurrent Neural Networks (Feb 2013), http://arxiv.org/abs/1211.5063, number: arXiv:1211.5063 arXiv:1211.5063 [cs]
30. programmersought.com: Chamfer Distance - Programmer Sought, https://programmersought.com/article/11413715914/#3D_16, uRL: https://programmersought.com/article/11413715914/#3D_16 Access Date: 2022-05-10
31. Qi, C.R., Su, H., NieBner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and Multi-view CNNs for Object Classification on 3D Data. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5648–5656. IEEE, Las Vegas, NV, USA (Jun 2016). https://doi.org/10.1109/CVPR.2016.609, http://ieeexplore.ieee.org/document/7780978/
32. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
33. Rukhovich, D., Vorontsova, A., Konushin, A.: ImVoxelNet: Image to Voxels Projection for Monocular and Multi-View General-Purpose 3D Object Detection (Oct 2021), http://arxiv.org/abs/2106.01178, number: arXiv:2106.01178 arXiv:2106.01178 [cs]
34. Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., Měch, R.: ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds (Sep 2020), http://arxiv.org/abs/2003.12181, number: arXiv:2003.12181 arXiv:2003.12181 [cs]
35. Shimon, U.: The interpretation of structure from motion. Proceedings of the Royal Society of London. Series B. Biological Sciences **203**(1153), 405–426 (Jan 1979). https://doi.org/10.1098/rspb.1979.0006, https://royalsocietypublishing.org/doi/10.1098/rspb.1979.0006
36. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view Convolutional Neural Networks for 3D Shape Recognition. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 945–953. IEEE, Santiago, Chile (Dec 2015). https://doi.org/10.1109/ICCV.2015.114, http://ieeexplore.ieee.org/document/7410471/
37. Su, J.C., Gadelha, M., Wang, R., Maji, S.: A Deeper Look at 3D Shape Classifiers. In: Leal-Taixé, L., Roth, S. (eds.) Computer Vision – ECCV 2018 Workshops, vol. 11131, pp. 645–661. Springer International Publishing, Cham (2019), http://link.springer.com/10.1007/978-3-030-11015-4_49, series Title: Lecture Notes in Computer Science
38. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018, vol. 11215, pp. 55–71. Springer International Publishing, Cham (2018), http://link.springer.com/10.1007/978-3-030-01252-6_4, series Title: Lecture Notes in Computer Science

39. Wang, T., Zhu, X., Pang, J., Lin, D.: FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection (Sep 2021), http://arxiv.org/abs/2104.10956, number: arXiv:2104.10956 arXiv:2104.10956 [cs]
40. Wang, X., Xu, Y., Xu, K., Tagliasacchi, A., Zhou, B., Mahdavi-Amiri, A., Zhang, H.: PIE-NET: Parametric Inference of Point Cloud Edges (Oct 2020), http://arxiv.org/abs/2007.04883, number: arXiv:2007.04883 arXiv:2007.04883 [cs]
41. Willis, K.D.D., Jayaraman, P.K., Lambourne, J.G., Chu, H., Pu, Y.: Engineering Sketch Generation for Computer-Aided Design. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 2105–2114. IEEE, Nashville, TN, USA (Jun 2021). https://doi.org/10.1109/CVPRW53098.2021.00239, https://ieeexplore.ieee.org/document/9523001/
42. Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B.: Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (Jan 2017), http://arxiv.org/abs/1610.07584, number: arXiv:1610.07584 arXiv:1610.07584 [cs]
43. Wu, R.: DeepCAD - code for our ICCV 2021 paper "DeepCAD: A Deep Generative Network for Computer-Aided Design Models" (Jun 2022), https://github.com/ChrisWu1997/DeepCAD, uRL: https://github.com/ChrisWu1997/DeepCAD Access Date: 2022-06-24
44. Wu, R., Xiao, C., Zheng, C.: DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6752–6762. IEEE, Montreal, QC, Canada (Oct 2021). https://doi.org/10.1109/ICCV48922.2021.00670, https://ieeexplore.ieee.org/document/9710909/
45. Xu, X., Peng, W., Cheng, C.Y., Willis, K.D., Ritchie, D.: Inferring CAD Modeling Sequences Using Zone Graphs. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6058–6066. IEEE, Nashville, TN, USA (Jun 2021). https://doi.org/10.1109/CVPR46437.2021.00600, https://ieeexplore.ieee.org/document/9577867/
46. Yagubbayli, F., Tonioni, A., Tombari, F.: LegoFormer: Transformers for Block-by-Block Multi-view 3D Reconstruction (Jun 2021), http://arxiv.org/abs/2106.12102, number: arXiv:2106.12102 arXiv:2106.12102 [cs]
47. Yan, X., Yang, J., Yumer, E., Guo, Y., Lee, H.: Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. Advances in neural information processing systems **29** (2016)
48. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4540–4549. IEEE, Seoul, Korea (South) (Oct 2019). https://doi.org/10.1109/ICCV.2019.00464, https://ieeexplore.ieee.org/document/9010395/
49. Yang, Y., Feng, C., Shen, Y., Tian, D.: FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 206–215. IEEE, Salt Lake City, UT (Jun 2018). https://doi.org/10.1109/CVPR.2018.00029, https://ieeexplore.ieee.org/document/8578127/
50. Zreik, M., van Hamersvelt, R.W., Wolterink, J.M., Leiner, T., Viergever, M.A., Isgum, I.: A Recurrent CNN for Automatic Detection and Classification of Coronary Artery Plaque and Stenosis in Coronary CT Angiography. IEEE Transactions on Medical Imaging **38**(7), 1588–1598 (Jul

2019). https://doi.org/10.1109/TMI.2018.2883807, https://ieeexplore.ieee.org/document/8550784/