Revolutionizing Computer-Aided Design Systems: CAD Sequence Inference from Product Image

Xingang Li
Walker Department of Mechanical Engineering
University of Texas at Austin
Austin, USA
xingang.li@utexas.edu

Zhenghui Sha
Walker Department of Mechanical Engineering
University of Texas at Austin
Austin, USA
zsha@austin.utexas.edu

I. Introduction

Computer-aided design (CAD) systems can significantly decrease design time by avoiding the need for labor-intensive manual drawings traditionally required [1]. Contemporary CAD systems such as Fusion 360, and SOLIDWORKS enable designers to create and modify CAD models through a sequence of CAD operations. However, in certain scenarios, the CAD model of a product may not be readily available due to various factors, including outdated documentation and the lack of digital records. Reverse engineering (RE) is employed to overcome these obstacles, utilizing measurement and analysis tools to reconstruct CAD models [2]. Integrating RE with CAD systems allows designers to leverage the advantages of existing products while incorporating their own innovative ideas and improvements. However, manual reconstruction of CAD models is labor-intensive and time-consuming, leading to the exploration of data-driven approaches, such as converting 3D point clouds into CAD models [3], [4].

Compared to point clouds, images are simpler to acquire. Thus, our question arises: Is it feasible to generate a CAD model using a single image? There is a scarcity of research on reconstructing 3D CAD models from images. We pioneer this area, and our objective is to generate a sequence of CAD operations based on a single image (referred to as "Image2CADSeq" hereafter for brevity). A CAD sequence offers advantages over 3D CAD models, enabling flexibility in modifying steps, and facilitating a better understanding of the historical process of the CAD model construction. Based on the initial findings of our ongoing study, we anticipate that the proposed method has the potential to revolutionize existing CAD systems by making the CAD model reconstruction process more accessible to a wider audience. This would enable both experienced and novice designers to actively contribute to the design, promoting design collaboration. Moreover, it has the potential to involve customers in the design process, fostering design democratization.

II. Methodology

A. Neural Network and Training

1) The Image2CADSeq Network: We have developed the Image2CADSeq network, illustrated in Figure 1, utilizing a target-embedding variational autoencoder (TEVAE) architecture [5]. The network employs a variational autoencoder (VAE) to encode the target objects (CAD sequences) into a latent space, and a separate encoder is used to regress the latent space using the input feature objects (i.e., images). For the VAE, we utilize a transformer-based VAE [3], while a typical ResNet18 [6] is employed for the encoder.

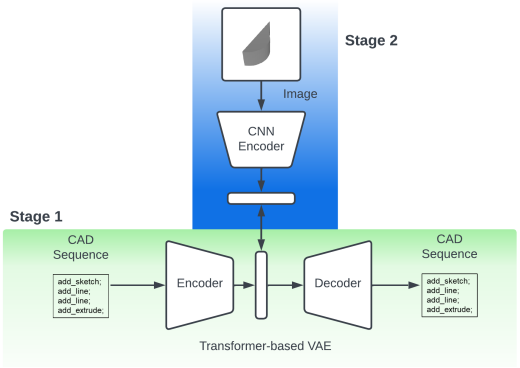

Fig. 1. The Image2CADSeq network and the two-stage training strategy

2) Two-Stage Training: We adopt a two-stage training approach [5]. In Stage 1, the VAE is independently trained. Subsequently, in Stage 2, the trained VAE's learning parameters are kept fixed, and the $Enc(\cdot)$ component is trained. By minimizing the Euclidean loss between the latent vectors obtained from the VAE and the embedding vectors generated by $Enc(\cdot)$, the $Enc(\cdot)$ module is trained using input images.

B. Fusion 360 Gallery Domain-Specific Language

We leverage the Fusion 360 Gallery domain-specific language (referred to as Gallery DSL hereafter for brevity) [7] for the CAD sequences. In each Gallery DSL command, there are two components: the command type and its corresponding parameters. However, the number of parameters varies for different commands, which presents challenges in devising a suitable vectorized design representation for the network. Drawing inspiration from the works [3], [8], we propose employing a fixed-dimensional vector that utilizes 10 parameters, as outlined in Table I, to encode each command.

TABLE I
REPRESENTATIVE PARAMETERS $t, I, x, y, \alpha, r, [I], d, O, s$

Parameter	Interpretation	Parameter Type	Value Range
t	Command type	Discrete	{0, 1, 2, 3, 4, 5, 6}
I	Identifier of sketch plane	Discrete	{0, 1, 2}
x	End point x	Continuous	[-1, 1]
y	End point y	Continuous	[-1, 1]
α	Sweep angle	Continuous	[-1, 0) or (0, 1] (*180)
r	Radius	Continuous	(0, 1]
$[I]$	Identifier of profile	Discrete	{**0**, 1, 2, ...}
d	Extrusion distance	Continuous	[-1, 0) or (0, 1]
O	Boolean operations	Discrete	{0, 1, 2, **3**}
s	Scale factor	Discrete	0~255 (e.g., **10**)

(1) t represents the command types. $0-4$ correspond to add_sketch, add_line, add_arc, add_circle, and $add\_extrude$, while 5 and 6 indicate the start and end of a command sequence (SOS) and (EOS), respectively.
(2) I indicates the sketch plane: "XY", "XZ", or "YZ".
(3,4) x and y represent the coordinates for Lines and Arcs (endpoint) or Circle (center point). The start point for Lines and Arcs is excluded from the design representation as it is derived from the preceding curve, ensuring connected curves.
(5) α represents the sweep angle of an Arc.
(6) r is the radius of a Circle.
(7) $[I]$ represents the profile index in the sketch.
(8) d is the signed distance of the extrusion depth.
(9) O indicates Boolean operations: join, cut, intersect, or add.
(10) s is a scale factor for scaling a CAD model.

There are different types of parameters: some are continuous values (e.g., coordinates x and y), while others are discrete values (e.g., the Boolean operation O). To handle both types of parameters uniformly, a common approach is to convert the continuous values into discrete values through quantization [3], [8]. To achieve this, we restrict the range of continuous values to a subset of $[-1, 1]$ (e.g., (0, 1] for radius). Then, we divide each range into 256 equal segments, allowing us to represent them as 8-bit integers ranging from 0 to 255. Especially, for the sweep angle α, we multiply it by 180 during interpretation. Notably, the scale factor s can be any non-negative continuous value, but for consistency, we limit it to 256 levels.

C. Vectorized Design Representation

A unified design representation for each Gallery command can be achieved by utilizing a fixed-dimensional vector. In this study, we employ a 7-dimensional vector $[t, I, x, y, \alpha, r, d]$ (highlighted in Table I) as part of our data synthesis pipeline, as described in Section II-D. When interpreting the 7-dimensional vectorized representation, we default $[I]$, O, and s to values of 0, 3, and 10, respectively. In future work, we intend to incorporate all parameters for a more comprehensive representation, thereby increasing the complexity of the synthesized data.

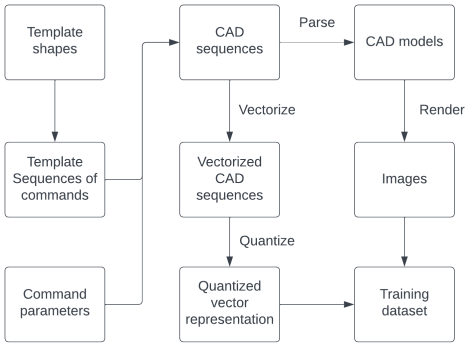

Fig. 2. Data synthesis pipeline

To ensure consistency across CAD programs with varying command sequence lengths, we address the need for a fixed total number of commands. For instance, a cylinder can be represented by five commands: $[5, 0, 3, 4, 6]$, while a triangular prism may require seven commands: $[5, 0, 1, 1, 1, 4, 6]$ (recall Table I). To achieve uniformity, we append 6 (i.e., EOS) to the end of the sequence until it reaches a fixed length (N_l). In this study, we set N_l to 10, ensuring it is greater than the longest sequence for the synthesized objects. For example, the command types of a triangular prism are represented as $[5, 0, 1, 1, 1, 4, 6, 6, 6, 6]$.

D. Automatic Data Synthesis Pipeline

Using the vectorized representation, we develop a data synthesis pipeline for training data, illustrated in Figure 2. The pipeline begins by preparing a collection of template shape primitives that adhere to the sketch-extrusion paradigm. For these template shapes, we formalize sequences of command types (e.g., add_arc, add_line) as templates. By combining the template sequences with the required parameters for each command, we generate complete CAD programs that can be parsed using the Fusion 360 Python API to create 3D CAD models automatically. Subsequently, we render the CAD models to obtain their corresponding images. The CAD programs are then vectorized to generate the vectorized representation described in Section II-C. An image (x) and its corresponding vector representation (y), both derived from the same CAD program, compose one data pair (x, y).

III. IMPLEMENTATION DETAILS AND RESULTS

A. Training Data Preparation

We employed two different approaches for dataset preparation: 1) Random synthesis: CAD programs were randomly synthesized by assigning command parameters within the predefined value range specified in Table I. For example, when generating an $add\_line(\cdot)$ command, we randomly selected a coordinate value of the endpoint from the range of $[-1, 1]$. 2) Synthesis with design rules: We established specific rules for parameter values. For instance, the extrusion depth of a circle was determined by the coordinates of its center point.

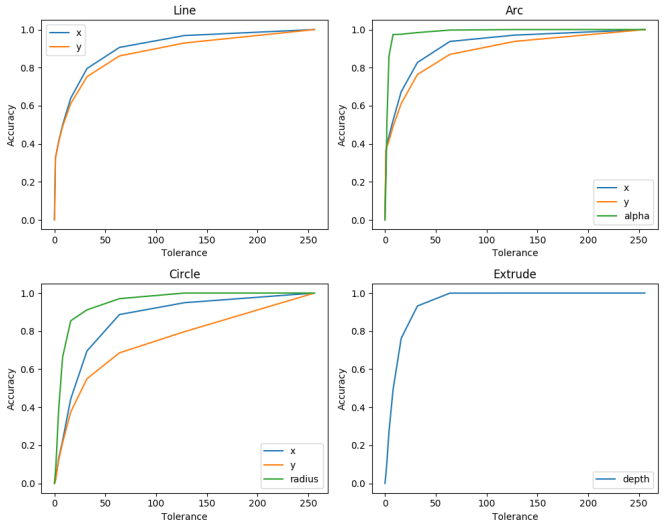

Fig. 3. Parameter accuracy VS. tolerance $(0-255)$ for Line, Arc, Circle, and Extrude commands

By incorporating design rules, we aimed to simulate how designers create designs, infusing purpose and intention into the synthesized data. Initially, a set of five shape primitives was prepared, and using each approach, we synthesized a total of 22,000 CAD models. For each synthesized model, we obtained corresponding images and vectorized design representations to form the training dataset.

B. Training Details

The training dataset was divided into three sets, namely the training set, validation set, and test set, with a ratio of 8:1:1, respectively. In Stage 1, we utilized the vector representations denoted as Y to train the VAE for 1000 epochs using a latent dimension of 256. For stage 2, we trained $Enc(\cdot)$ utilizing the pre-trained ResNet18 model [6] for 50 epochs. The training was conducted using the Adam optimizer with a learning rate of 0.0001 and a batch size of 128.

C. Preliminary Results

We conducted a comparison between two datasets in terms of sequence accuracy for CAD sequence generation. The dataset that incorporated design rules yielded significantly higher sequence accuracy compared to the dataset without rules (**0.961** VS. **0.432**) on the unseen test set data. This finding highlights the importance of integrating design rules into the data synthesis process, as it enhances data quality and subsequently improves the performance of the network.

Additionally, we analyzed the accuracy of command parameters in relation to parameter tolerance, as depicted in Figure 3. With a tolerance of 3, the average parameter accuracy for all parameters, except t and I (for which tolerance was set to 0), was found to be 0.446. While it is impractical to have excessively large tolerance values, Figure 3 demonstrates that the Image2CADSeq network's ability to predict the correct parameters varies for different parameters. Notably, the network exhibits poor performance in predicting the coordinates of the circle's center. This issue is closely linked to the design rules, as we randomly selected a center point for circles during the data synthesis process.

IV. Conclusion

The main objective of the Image2CADSeq network is to generate a CAD sequence, consisting of command types and associated parameters, based on an input image. For training purposes, we utilized synthesized data representing simple shape primitives. Although the parameter prediction accuracy is less than 50%, the network demonstrates a promising accuracy of up to 96% in predicting the sequence of command types. This indicates the network's potential to accurately predict the CAD sequence given a single image. In future research, we plan to evaluate the network's performance on more complex shapes to further assess its capabilities. We will also explore different network architectures and enhance the quality of the training data by incorporating design rules into the data synthesis process.

The proposed approach holds significant potential to bring about transformative changes in existing CAD systems, revolutionizing the product development cycle. Moreover, it has the capacity to democratize the CAD model reconstruction process, allowing individuals with limited experience or expertise to actively participate. By removing barriers, it can also facilitate customer engagement in design activities, promoting the democratization of design.

Acknowledgment

The authors gratefully acknowledge the financial support from the National Science Foundation through award 2207408.

References

[1] Rosato, D., and Rosato, D., 2003. "5 - computer-aided design". In *Plastics Engineered Product Design*, D. Rosato and D. Rosato, eds. Elsevier Science, Amsterdam, pp. 344–380.

[2] Buonamici, F., Carfagni, M., Furferi, R., Governi, L., Lapini, A., and Volpe, Y., 2018. "Reverse engineering modeling methods and tools: a survey". *Computer-Aided Design and Applications,* ***15***(3), pp. 443–464.

[3] Wu, R., Xiao, C., and Zheng, C., 2021. "Deepcad: A deep generative network for computer-aided design models". In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6772–6782.

[4] Uy, M. A., Chang, Y.-Y., Sung, M., Goel, P., Lambourne, J. G., Birdal, T., and Guibas, L. J., 2022. "Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11850–11860.

[5] Li, X., Xie, C., and Sha, Z., 2022. "A predictive and generative design approach for three-dimensional mesh shapes using target-embedding variational autoencoder". *Journal of Mechanical Design,* ***144***(11), p. 114501.

[6] He, K., Zhang, X., Ren, S., and Sun, J., 2016. "Deep residual learning for image recognition". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[7] Willis, K. D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J. G., Solar-Lezama, A., and Matusik, W., 2021. "Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences". *ACM Transactions on Graphics (TOG),* ***40***(4), pp. 1–24.

[8] Carlier, A., Danelljan, M., Alahi, A., and Timofte, R., 2020. "Deepsvg: A hierarchical generative network for vector graphics animation". *Advances in Neural Information Processing Systems,* ***33***, pp. 16351–16361.