JMST Advances (2023) 5:113–119
https://doi.org/10.1007/s42791-023-00059-5

Online ISSN 2524-7913
Print ISSN 2524-7905

REVIEW

Reconstruction of design history of 3D CAD models using deep learning: research trends

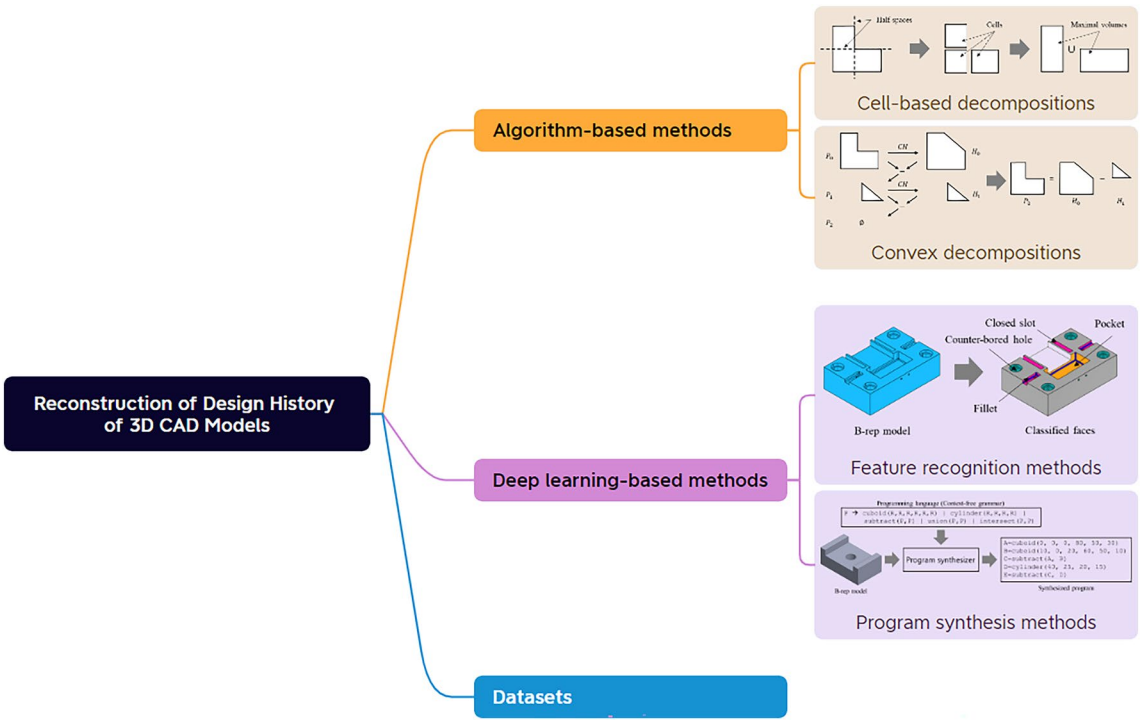
Byung Chul Kim[1]

Received: 20 September 2023 / Revised: 23 November 2023 / Accepted: 24 November 2023 / Published online: 9 December 2023
© The Korean Society of Mechanical Engineers 2023

Abstract

Design history reconstruction refers to the process of identifying the features that comprise a computer-aided design (CAD) model, determining the parameter values associated with these features, and establishing the order in which the features were applied. This is necessary when working with a CAD model that lacks a design history. While commercial CAD programs typically support design history as their primary representation, it is common for the design history to be lost in many cases due to limitations in standard CAD file formats or for the protection of design knowledge. This loss of design history can make it challenging to modify CAD models and can result in higher product design costs. Research on design history reconstruction has been ongoing since the late 1980s to tackle this issue. However, previous studies have struggled to find solutions that can be widely applied. Nevertheless, recent advancements in deep learning have prompted researchers to explore the use of deep learning techniques for design history reconstruction. This article presents several such research endeavors, along with a discussion of pending challenges and directions for future research.

Graphical abstract



Keywords CAD model · Deep learning · Design history · Reconstruction

Extended author information available on the last page of the article

1 Introduction

In modern manufacturing, computer-aided design (CAD) models have become a key component for integrating design, analysis, and production. Designers employ CAD to create three-dimensional (3D) models of products, utilizing these models for analysis and process planning, allowing them to preemptively identify potential product issues. Subsequently, the 3D CAD model undergoes iterative modifications to address issues or achieve the optimal design. Furthermore, the life cycle of the final CAD model does not conclude at this point; it is frequently reused in other designs with minor modifications. This process is made feasible through the parametric representation of CAD models.

3D CAD models are represented as a procedural model of features. A feature is a geometric shape with engineering meaning, defined by parameters. For example, a drilling feature not only has a cylindrical shape but also includes the meaning of the drilling operation, with parameters such as drill diameter and drilling depth. Designers apply various features in a specific sequence to define the final shape. Consequently, the order in which features are applied, known as the design history, is crucial. When modifying a CAD model, designers modify the parameter values of the features. Subsequently, by reapplying the features according to the design history, a modified shape is generated [1].

Unfortunately, incorporating features and design history into released 3D CAD models is challenging. First, the widely recognized 3D CAD file exchange format, initial graphics exchange specification (IGES), lacks support for features and design histories. Additionally, while the international standard ISO 10103 [2], known as standard for the exchange of product model data (STEP), does support these features, commercial CAD systems do not yet fully support this standard. Consequently, only shapes represented by boundary representation (B-rep) models can currently be used. Another reason that enterprises intentionally exclude design history from CAD models shared with contractors is to safeguard proprietary design knowledge. Although shapes can be reconstructed through reverse engineering, only the final shape can be obtained, making CAD model modification extremely difficult when design history is excluded. This results in prolonged product development cycles and escalated costs.

From a manufacturing perspective, converting the design features of the reconstructed design history into machining features enables integration between CAD and computer-aided manufacturing (CAM). It would also be possible to utilize the additive and subtractive parts of the design features to enable hybrid additive–subtractive manufacturing.

To address these issues, extensive research efforts have been conducted since the late 1980s aimed at reconstructing the design history of 3D CAD models [3]. However, practical solutions have not been proposed due to various difficulties. Nevertheless, given the significant advancements witnessed in other fields through the application of deep learning, endeavors are being made to harness deep learning techniques in tackling outstanding challenges in the domain of design history reconstruction. This study will explore the latest research trends in design history reconstruction. It will commence by examining research predating the era of deep learning, followed by summarizing methodologies utilizing deep learning. Furthermore, it will examine published datasets required for deep learning applications. Finally, the study will discuss pending issues and outline potential directions for future research.

2 Algorithm-based methods

The field of design history reconstruction for 3D CAD models has a long history. The primary aim of this research field is to reconstruct design histories for CAD models without design histories. CAD models without design history can be represented through various models, including voxel models and mesh models, but boundary representation (B-rep) models are commonly used as input shapes. Attempts have been made to convert these into constructive solid geometry (CSG) models.

B-rep models are the most commonly used models for representing shapes in CAD. They explicitly define shapes using geometric entities such as points, curves, and surfaces, as well as topological entities such as vertices, edges, and faces. In contrast, CSG models employ Boolean operations of parametrized simple shapes such as spheres, cylinders, and cubes. Instead of explicit shapes, CSG models contain construction history and parameter values. Converting B-rep models into CSG models allows for the reconstruction of design histories. However, this process is more challenging compared to converting CSG models to B-rep models. Major methods for achieving this include cell-based decomposition and convex decomposition.

In cell-based decomposition, half-spaces that encompass the faces of the B-rep model are used to split a shape into simple cells, which are subsequently reassembled to meet specified conditions for maximum volume. In this method, the design history of the original shape is represented as a Boolean union of simple shapes. Shapiro and Vossler [4] were the first to propose a way to convert a B-rep model to a CSG model by dividing space using half-spaces. Sakurai [5, 6] proposed a maximal volume decomposition method where, for faces sharing convex edges, the shape is decomposed into cells using half-spaces encompassing each face, then these cells are recombined to find a convex

shape is as large and simple as possible. Kim and Mun [7] proposed a method for decomposing volumes in sequential iterative order using various volume decomposition operations. Figure 1a shows an example of maximum volume decomposition.

Convex decomposition is a method of sequentially decomposing an original shape using convex hulls and delta volumes. In this method, the design history of the original shape is represented by a binary tree of Boolean unions and subtractions of the decomposed volumes. Tang and Woo [8] proposed an alternating sum of volumes (ASV) decomposition method using convex hulls of the B-rep model. In this method, for an original shape P_0, the convex hull $H_0 = CH(P_0)$ is calculated. This is followed by calculation of the difference $P_1 = H_0 - P_0$. This process is repeated until $P_{i+1} = H_i - P_i$ becomes ∅. The original shape P_0 is then expressed as $\sum (-1)^i H_i$. Refining further, the process can be expressed as an operation of subtracting simple shapes from the original shape. Figure 1b shows an example of the ASV decomposition method. Kim [9] proposed an alternating sum of volumes with partitioning (ASVP) decomposition method that addresses the problems of ASV, along with a form-feature decomposition (FFD) method that further develops the approach.

Several other modified and improved methods have been attempted, evolving into feature recognition techniques. Nevertheless, none of these methods have achieved widespread applicability. Moreover, algorithm-based approaches require long processing times when dealing with complex shapes and are limited in their ability to recognize parameters for the basic shapes that constitute CSG models.

3 Deep learning-based methods

3.1 Feature recognition methods

Algorithm-based methods have evolved from studies focusing on the conversion of B-rep models into CSG models to research on feature recognition. Therefore, investigating how current deep learning-based feature recognition is conducted is meaningful. Strictly speaking, current deep learning-based methods for feature recognition are not truly recognizing features. As shown in Fig. 2, most studies aim to classify the individual faces of a B-rep model based on their feature types. Consequently, from a technical standpoint, these methods amount to face classification based on features.

In the early studies [10–14], researchers primarily employed artificial neural networks (ANN). In these studies, the faces and edges of a B-rep model were converted into attributed graphs. Subsequently, heuristics were utilized to restructure the graph into a format suitable for input into an ANN. However, due to limitations in

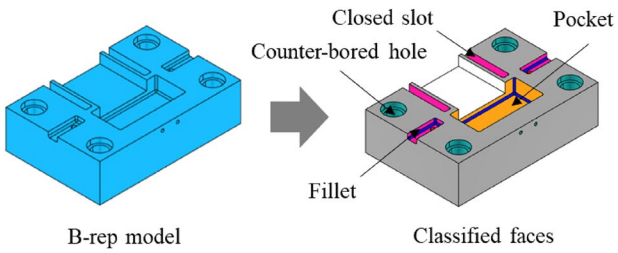


Fig. 2 Example of feature recognition

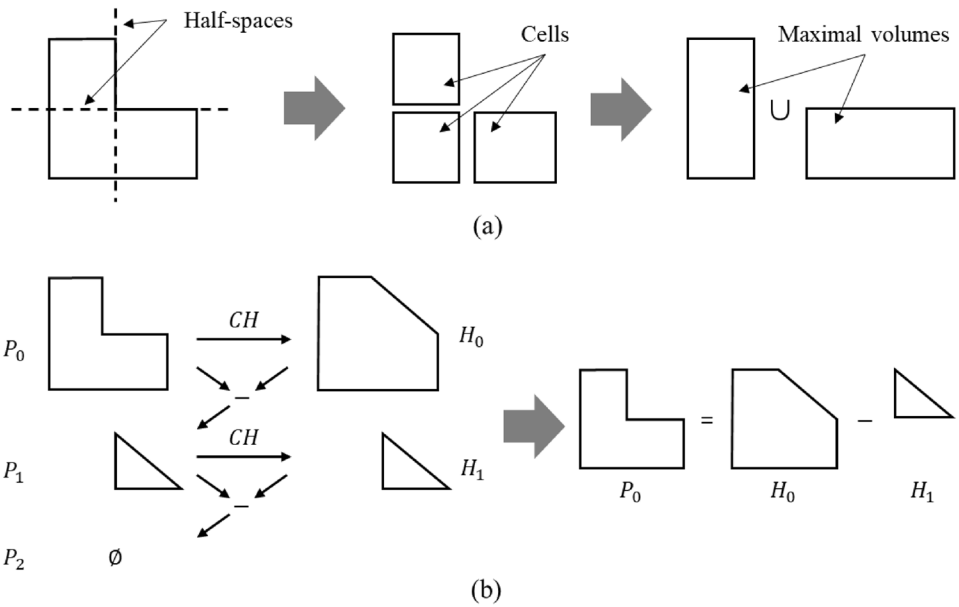


Fig. 1 **a** maximum volume decomposition and **b** ASV decomposition

computer performance at that time, these methods were not feasible for large models.

With the advent of deep learning, attempts were made to apply these methods to feature recognition. However, there was no effective method for using B-rep models as inputs for neural networks, so these models had to be approximated into a format suitable for deep learning. Typically, shapes were approximated using voxel models [15] and point clouds [16]. Nevertheless, approximated models cannot precisely represent three-dimensional shapes, and their accuracy varies with resolution. Additionally, they fail to capture the geometric properties of the shape and struggle to represent the topological structures of the B-rep model. Most importantly, handling these models requires significant memory resources, making high-resolution approximations a challenge.

Recently, methods that apply B-rep models to deep learning without the need for three-dimensional shape approximation have been proposed. Cao et al. [17] proposed a method for using B-rep models directly. In this approach, the faces and edges constituting a B-rep model are represented as a face adjacency graph, and features are identified using a graph neural network. However, this technique is limited to shapes composed of planes. To address this limitation, UV-Net [18] and Hierarchical CADNet [19] focus on learning geometric information for curves and surfaces. UV-Net approximates the faces and edges of the B-rep model into two-dimensional grids and one-dimensional grids in UV space, respectively. Feature vectors for faces and edges are then extracted using a convolutional neural network (CNN). The adjacency relationships among these faces and edges are represented as a graph, with a graph neural network used for face classification. Similarly, hierarchical CADNet represents the B-rep model as a face adjacency graph, projecting it from the B-rep model dimension to a unit face dimension to extract curve features. Face classification is then achieved using a graph convolution network (GCN). These methods convert the data structure of the B-rep model into a face adjacency graph, resulting in the loss of some topological position information with neighboring entities. Lambourne et al. [20] proposed an alternative approach for face classification. They utilized feature vectors of neighboring faces and edges based on co-edges. In this study, adjacency relationships are represented as a matrix to represent the connections between topological entities. The concept of topological walk is adopted to represent the features of neighboring entities as a single feature matrix. Subsequently, multi-layer perceptrons (MLP) are utilized for face classification. Cha and Kim [21] introduced a method where, after face classification, connected-component analysis is used to identify sets of faces included in the same features.

Feature recognition methods exhibit high performance, with accuracy exceeding 95%. However, they can only classify individual faces based on features, and are unable to find sets of faces that constitute a feature. Further, they are incapable of recognizing the design history, which is the order of the features, and the parameters for defining features.

3.2 Program synthesis methods

Program synthesis is a method in which elements of the underlying programming language are combined to automatically find a program that can fulfill a specific purpose. As shown in Fig. 3, when program synthesis is applied to design history reconstruction, the simple shapes of the CSG, Boolean operations involving these shapes, features, parameter values, and so on, are treated as elements of the programming language. These elements are then combined to synthesize a program capable of generating the desired shape. This is made possible because CAD models are procedural models. The challenge in program synthesis lies in the vast search space for synthesizable programs. Therefore, the key to successful program synthesis is to combine programming language elements effectively and find a program that can achieve the intended purpose within a reasonable amount of time. Accordingly, various studies have proposed methods to address it.

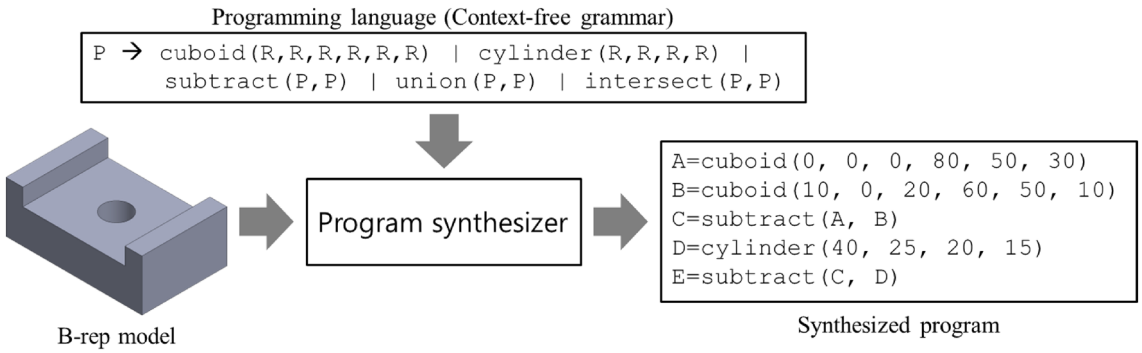


Fig. 3 Example of program synthesis

Du et al. [22] proposed InverseCSG as a method for generating CSG programs from 3D shapes. Whereas this method does not employ deep learning, a method for finding CSG programs efficiently in the search space using various heuristics was proposed. Sharma et al. [23] proposed a CSG-Net for reconstructing CSG models from voxel models using the encoder–decoder model widely used in language models. In CSGNet, the encoder uses a CNN to extract feature vectors from the voxel model. The decoder then outputs a CSG program from the feature vectors using a recurrent neural network (RNN) consisting of gated recurring units (GRUs). Kania et al. [24] proposed a UCSG-Net similar to CSGNet. However, UCSG-Net uses shapes represented as signed distance fields (SDF) as input. Tian et al. [25] proposed a model consisting of a program generator and a program executive. The program generator serves as the encoder which synthesizes the shape-generating program, while the program executor serves as the decoder which predicts the result of the synthesized program. The models for both programs are represented using a CNN and an RNN. Ellis et al. [26] proposed a universal method for program synthesis wherein the program is partially written, then the execution result of the partially written program is used to find the target program. This method was demonstrated by reconstructing a CSG model. In this method, search is not performed until the program is fully synthesized. Instead, if the desired result is not yielded from the partially synthesized program, the search is not continued. This increases search speed. A reinforcement learning technique using Monte Carlo search was applied to writing and executing the program to improve speed.

Program synthesis methods are the most promising for achieving the goal of design history reconstruction as they can find design history, features, and parameter values from the input shape. However, existing studies have primarily used voxel models or images as input shapes, which restricts their capacity to fully reconstruct the original shape.

4 Datasets

Training datasets play a key role in achieving maximum performance with deep learning. While datasets for image and language tasks are abundant, collecting datasets for CAD models is time-consuming, and often these datasets do not align with the desired model representation. Consequently, many studies opt to synthetically create CAD model datasets while recording the generative process. For CSGNet [23], datasets were generated using randomly created shapes composed of spheres, cubes, and cylinders. However, due to their random generation, these datasets contain numerous unrealistic shapes.

In hierarchical CADNet [19], an MFCAD + + dataset was artificially generated and used. The MFCAD + + dataset comprises 59,655 CAD models containing 24 machining features. For MFCAD + +, several rules were applied to give the resemblance of CAD models created by humans. Nonetheless, this dataset is limited to machining features, and the shapes are relatively simple. Additionally, since the design history is not separately provided, researchers must generate it following the proposed method.

Fusion 360 Gallery [27] provides a dataset with design history expressed through sketch commands and extrusion commands for CAD models created by humans. This dataset encompasses 8,625 shapes, and it also provides Fusion 360 Gym, suitable for reinforcement learning environments. However, it primarily features a single type of feature—extrusion. Various other CAD model datasets, including the ABC dataset [28], FeatureNet dataset [15], and FabWave3D [29], typically provide labeling data for shape and face classifications but lack comprehensive information required for design history reconstruction.

5 Limitations and future research directions

The design history reconstruction methods presented have several limitations. First, there is no unique design history for the same shape, so the results of reconstruction using each method may be different. To overcome this challenge, a method for assessing the quality of the design history is needed. Second, existing studies have focused on elementary shapes such as spheres and boxes when reconstructing design histories. Consequently, there is a pertinent need to explore methodologies that facilitate the reconstruction of features and sketch commands at a level akin to established commercial CAD programs. Third, the representation of relative feature positions through references to faces, edges, and vertices is essential. While this is common practice in commercial CAD programs, previous methods have relied on absolute positions. Finally, the lack of a dedicated dataset for deep learning design history reconstruction is a significant challenge. This lack precludes a standardized basis for evaluating and comparing the results of different studies. The construction of a standardized dataset is, therefore, essential to improve the reliability and comparability of results in this area.

6 Conclusion

As deep learning continues to advance, solutions to the challenge of design history reconstruction are steadily progressing. Currently, methods that combine program synthesis and reinforcement learning seem the most promising. However, there have been no studies that employ this approach to reconstruct design history for a B-rep model.

Hence, there is a need to address this gap and find a solution to this problem. Upon implementation, design history reconstruction will reduce product development time and costs significantly. It will play a crucial role in integrating CAD with CAM or enabling hybrid additive–subtractive manufacturing.

Acknowledgements This work was supported by the Korea Evaluation Institute of Industrial Technology (KEIT) Grant funded by the Korean Government through Ministry of Trade, Industry and Energy (MOTIE) under Project RS-2022-00143813.

Funding Korea Evaluation Institute of Industrial Technology, RS-2022-00143813, Byung Chul Kim.

Declarations

Conflict of interest None declared.

References

1. M. J. Pratt, Extension of ISO 10303, the STEP standard, for the exchange of procedural shape models, in *Proceedings of the Shape Modeling Applications 2004*, Genova, Italy (2004)
2. M.J. Pratt, B.D. Anderson, T. Ranger, Towards the standardized exchange of parameterized feature-based CAD models. Comput. Aided Des. **37**(12), 1251–1265 (2005)
3. J. Han, M.J. Pratt, W.C. Regli, Manufacturing feature recognition from solid models: a status report. IEEE Trans. Robot. Autom. **16**(6), 782–796 (2000)
4. V. Shapiro, D.L. Vossler, Construction and optimization of CSG representations. Comput. Aided Des. **23**(1), 4–20 (1991)
5. H. Sakurai, Volume decomposition and feature recognition: part 1—polyhedral objects. Comput. Aided Des. **27**(11), 833–843 (1995)
6. H. Sakurai, P. Dave, Volume decomposition and feature recognition, part II: curved objects. Comput. Aided Des. **28**(6–7), 519–537 (1996)
7. B.C. Kim, D. Mun, Feature-based simplification of boundary representation models using sequential iterative volume decomposition. Comput. Graph. **38**, 97–107 (2014)
8. K. Tang, T. Woo, Algorithmic aspects of alternating sum of volumes. Part 1: data structure and difference operation. Comput. Aided Des. **23**(5), 357–366 (1991)
9. T.S. Kim, Recognition of form features using convex decomposition. Comput. Aided Des. **24**(9), 461–476 (1992)
10. S. Prabhakar, M.R. Henderson, Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. Comput. Aided Des. **24**(7), 381–393 (1992)
11. K. Nezis, G. Vosniakos, Recognizing 212D shape features using a neural network and heuristics. Comput. Aided Des. **29**(7), 523–539 (1997)
12. L. Ding, Y. Yue, Novel ANN-based feature recognition incorporating design by features. Comput. Ind. **55**(2), 197–222 (2004)
13. V. Sunil, S. Pande, Automatic recognition of machining features using artificial neural networks. Int. J. Adv. Manuf. Technol. **41**(9–10), 932–947 (2009)
14. C. Yeo, B.C. Kim, S. Cheon, J. Lee, D. Mun, Machining feature recognition based on deep neural networks to support tight integration with 3D CAD systems. Sci. Rep. **11**(1), 22147 (2021)
15. Z. Zhang, P. Jaiswal, R. Rai, FeatureNet: Machining feature recognition based on 3D convolution neural network. Comput. Aided Des. **101**, 12–22 (2018)
16. Y. Ma, Y. Zhang, X. Luo, Automatic recognition of machining features based on point cloud data using convolution neural networks, in *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*, Wuhan, China (2019)
17. W. Cao, T. Robinson, Y. Hua, F. Boussuge, A.R. Colligan, W. Pan, Graph representation of 3D CAD models for machining feature recognition with deep learning, in *Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Online (2020)
18. P.K. Jayaraman, A. Sanghi, J. G. Lambourne, K.D.D. Willis, T. Davies, H. Shayani, N. Morris, UV-Net: Learning from boundary representations, in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA (2021)
19. A.R. Colligan, T.T. Robinson, D.C. Nolan, Y. Hua, W. Cao, Hierarchical CADNet: learning from B-Reps for machining feature recognition. Comput. Aided Des. **147**, 103226 (2022)
20. J. G. Lambourne, K.D.D. Willis, P.K. Jayaraman, A. Sanghi, P. Meltzer, H. Shayani, BRepNet: a topological message passing system for solid models, in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA (2021)
21. M.H. Cha, B.C. Kim, Machining feature recognition using BRepNet, in *Proceedings of the 9th International Conference on Manufacturing, Machine Design and Tribology*, Jeju, Korea (2023)
22. T. Du, J.P. Inala, Y. Pu, A. Spielberg, A. Schulz, D. Rus, A. Solar-Lezama, W. Matusik, InverseCSG: automatic conversion of 3D models to CSG trees. ACM Trans. Graph. **37**(6), 213 (2018)
23. G. Sharma, R. Goyal, D. Liu, E. Kalogerakis, S. Maji, CSGNet: Neural shape parser for constructive solid geometry, in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, USA (2018)
24. K. Kania, M. Zieba, T. Kajdanowicz, UCSG-NET-unsupervised discovering of constructive solid geometry tree. Adv. Neural. Inf. Process. Syst. **33**, 8776–8786 (2020)
25. Y. Tian, A. Luo, X. Sun, K. Ellis, W.T. Freeman, J.B. Tenenbaum, J. Wu, Learning to infer and execute 3D shape programs, in *Proceeding of the 7th International Conference on Learning Representations*, New Orleans, USA (2019)
26. K. Ellis, M. Nye, Y. Pu, F. Sosa, J. Tenenbaum, A. Solar-Lezama, Write, execute, assess: Program synthesis with a REPL, in *Advances in Neural Information Processing Systems*, 32 (2019)
27. K.D.D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J.G. Lambourne, A. Solar-Lezama, W. Matusik, Fusion 360 gallery: a dataset and environment for programmatic CAD construction from human design sequences. ACM Trans. Graph. **40**(4), 54 (2021)
28. S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, D. Panozzo, Abc: a big cad model dataset for geometric deep learning, in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, USA (2019)
29. A. Angrish, B. Craver, B. Starly, "FabSearch": a 3D CAD model-based search engine for sourcing manufacturing services. J. Comput. Inf. Sci. Eng. **19**(4), 041006 (2019)

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Byung Chul Kim received the B.S. in mechanical engineering from Korea University, Seoul, Republic of Korea, in 2001 and the M.S. and Ph.D. degrees in mechanical engineering from KAIST, Daejeon, Republic of Korea, in 2003 and 2008, respectively. He is currently an Associate Professor at the school of mechanical engineering, Korea University of Technology and Education. His research interests are computer-aided design, computer-aided manufacturing, point cloud and mesh processing, artificial intelligence-based design, product data modeling and exchange, and other related areas. His application domains are all industries related to mechanical engineering such as automobile, shipbuilding, power plant, aircraft, and semiconductor.

Authors and Affiliations

Byung Chul Kim[1]

✉ Byung Chul Kim
mir7942@koreatech.ac.kr

[1] School of Mechanical Engineering, Korea University of Technology and Education, Cheonan, Republic of Korea