



MeshCNN-based BREP to CSG conversion algorithm for 3D CAD models and its application

Yue-Tong Luo¹ · Hua Du^{2,3} · Yi-Man Yan⁴

Received: 10 February 2022 / Revised: 8 May 2022 / Accepted: 19 June 2022 / Published online: 6 July 2022

© The Author(s), under exclusive licence to China Science Publishing & Media Ltd. (Science Press), Shanghai Institute of Applied Physics, the Chinese Academy of Sciences, Chinese Nuclear Society 2022

Abstract In the field of neutronics analysis, it is imperative to develop computer-aided modeling technology for Monte Carlo codes to address the increasing complexity of reactor core components by converting 3D CAD model (boundary representation, BREP) to MC model (constructive solid geometry, CSG). Separation-based conversion from BREP to CSG is widely used in computer-aided modeling MC codes because of its high efficiency, reliability, and easy implementation. However, the current separation-based BREP-CSG conversion is poor for processing complex CAD models, and it is necessary to divide a complex model into several simple models before applying the separation-based conversion algorithm, which is time-consuming and tedious. To avoid manual segmentation, this study proposed a MeshCNN-based 3D-shape segmentation algorithm to automatically separate a complex model. The proposed 3D-shape segmentation algorithm was combined with separation-based BREP-CSG conversion algorithms to directly convert complex models.

The proposed algorithm was integrated into the computer-aided modeling software cosVMPT and validated using the Chinese fusion engineering testing reactor model. The results demonstrate that the MeshCNN-based BREP-CSG conversion algorithm has a better performance and higher efficiency, particularly in terms of CPU time, and the conversion result is more intuitive and consistent with the intention of the modeler.

Keywords BREP to CSG conversion · Computer-aided modeling · cosVMPT · Intelligent pre-segmentation · MeshCNN

1 Introduction

The first 3D boundary representation (BERP) to constructive solid geometry (CSG) conversion algorithm is an improved 2D ASV [1] algorithm. Although the improved ASV is easy to implement, its convergence cannot be guaranteed and it is only suitable for converting polyhedrons. Shapiro [2, 3] presented a solution to the BERP-CSG conversion for 3D solids in the 1990s, in which they solved a general conversion problem from BREP to half-space-based CSG. Shapiro proposed three concepts for generalizing the BERP-CSG algorithm: canonical intersection, a describable theorem based on half-spaces of boundaries, and half-space separation. The proposed algorithm can be used to construct half-space-based CSGs for solids bound by quadric surfaces. Moreover, half-space domination was proposed to simplify the CSG representation of a solid. Suzanne [4] proposed the BHC algorithm based on the fundamental work of Shapiro et al., who presented an effective representation method for an

This work was supported by the National Key R&D Program of China (Nos. 2019YFE03110000 and 2017YFE0300501) and the Chinese National Natural Science Foundation (No. 11775256).

✉ Yue-Tong Luo
ytluo@hfut.edu.cn

¹ Hefei University of Technology, Hefei 230009, China

² University of Science and Technology of China, Hefei 230027, China

³ Institute of Plasma Physics, Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, China

⁴ State Power Investment Corporation Research Institute, National Energy Key Laboratory of Nuclear Power Software, Beijing 102209, China

intermediate model that avoids the extra computation of computing a resultant solid after each pass of dominating half-space factorization. Additionally, half-spaces are factored one at a time, avoiding the additional post-processing computation required to minimize the resultant CSG-tree. Shapiro and Suzanne laid the theoretical foundation for 3D BREP-CSG conversion.

Monte Carlo (MC) particle transport codes [5] are widely used in the field of nuclear analysis, and require a 3D CSG model as the input. With the increasing complexity of nuclear facilities, it is necessary to use CAD software for modeling; however, the resulting model is the BREP model. Therefore, it is crucial to develop computer-aided modeling technologies to satisfy the conversion demands of CAD models (BREPs) to MC models (CSGs). To conduct this conversion, Luo [6] proposed a separation-based BREP-CSG conversion algorithm based on the theories of Shapiro and Suzanne. The core idea is to decompose a complex CAD model into a set of simple convex solids by using Boolean operations. The corresponding convex solids are represented by half-spaces of the boundary surfaces. Separation-based BREP-CSG conversion is reliable and easily implemented, resulting in software development for computer-aided modeling, including MCAM [7–9] and McCAD [10–12].

However, separation-based BREP-CSG conversion relies heavily on a large number of Boolean operations because it is time consuming and error-prone, particularly for models with complex structures. Additionally, the likely occurrence of excessive decomposition reduces the conversion efficiency and increases the computation time of the MC code. Luo [13] proposed feature-based BREP-CSG conversion to optimize BREP-CSG conversion, the basic idea of which is to identify the predefined features of a model using feature recognition technology. These features were used as tree nodes, and a separation-based BREP-CSG conversion was employed to convert each feature. Unfortunately, the model characteristics are inconspicuous or features are difficult to define. This limits the applicability of the proposed algorithm. Other optimization techniques [12] have been proposed to improve conversion efficiency, including replacing Boolean operations with triangle collisions, adding an auxiliary splitting surface algorithm to reduce the generation of irregular solids, and introducing a priority-sorting algorithm for splitting surfaces. Although the optimized conversion algorithm improved the conversion efficiency and stability, it required rules of thumb summarized by users. These rules frequently rely on low-level geometric elements, such as edges and faces, without considering the high-level features of the 3D models. Moreover, owing to the limitations of these rules, it is extremely difficult, or even impossible, to find a universal rule for all models.

Therefore, it is necessary to manually preprocess a complex model before conversion to avoid errors, particularly when converting a model with highly complex structures, such as the Chinese fusion engineering testing reactor (CFETR) [14, 15]. Figure 1 shows diagrams of the complex models. Relatively regular models are obtained through manual preprocessing and can be converted using computer-aided modeling software (e.g., cosVMPT). However, manual model preprocessing is time-consuming and tedious, and it is imperative to develop model preprocessing technology.

Manual model preprocessing decomposes a complex model into relatively regular sub-models according to the intention of the modeler. Generally, this process can be categorized as a 3D shape segmentation. To date, several studies have been conducted on 3D-shape segmentation [16–20]. Compared with other methods, deep learning-based segmentation has been widely studied owing to its effectiveness (see Sect. 2). Considering the high similarity between components, as shown in Fig. 2a, the vacuum vessel (VV) shares similar component structures with the thermal shield (TS). Moreover, there are some components in the other sectors, as shown in Fig. 2b. Therefore, considering the characteristics of a fusion model, deep learning-based 3D-shape segmentation can play a significant role in 3D model preprocessing. Accordingly, an intelligent pre-segmentation algorithm is proposed to learn preprocessing rules by inputting pre-segmented cases, with the learned rules then used to guide complex model segmentation.

The cosVMPT [21, 22] is a visual modeling platform that converts CAD models into MC models and utilizes OpenCascade (OCC) as a CAD geometry engine. The FreeCAD framework is a graphical user interface, and VTK provides data visualization. The cosVMPT was successfully applied to complex CFETR fusion devices comprising tens of thousands of solids (Fig. 3), including the plasma chamber, blanket, divertor, cryostat, VV, TS, toroidal field coil (TFC), poloidal field coil, and central solenoid. An intelligent pre-segmentation algorithm was proposed to optimize the BREP-CSG conversion algorithm to prevent the need for manual intervention in model preprocessing and improve the conversion performance. Intelligent pre-segmentation is a combination of deep learning-based 3D model segmentation with separation-based BREP-CSG conversion. The 3D model segmentation algorithm was used to learn the preprocessing rules for the segmentation of the CAD model and intermediate model generation, according to the intention of the modeler. A separation-based BREP-CSG conversion algorithm was used to convert the intermediate models to optimize BREP-CSG conversion. An intelligent pre-segmentation algorithm was applied to cosVMPT and applied practically.

Fig. 1 (Color online) Diagrams of the conversion of certain complex models

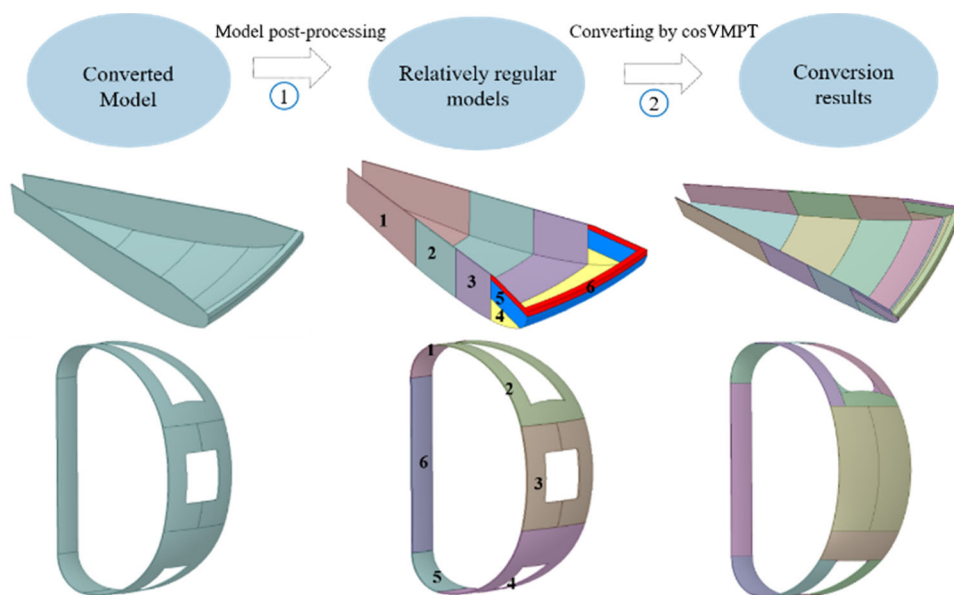


Fig. 2 (Color online) A fusion component with high similarity

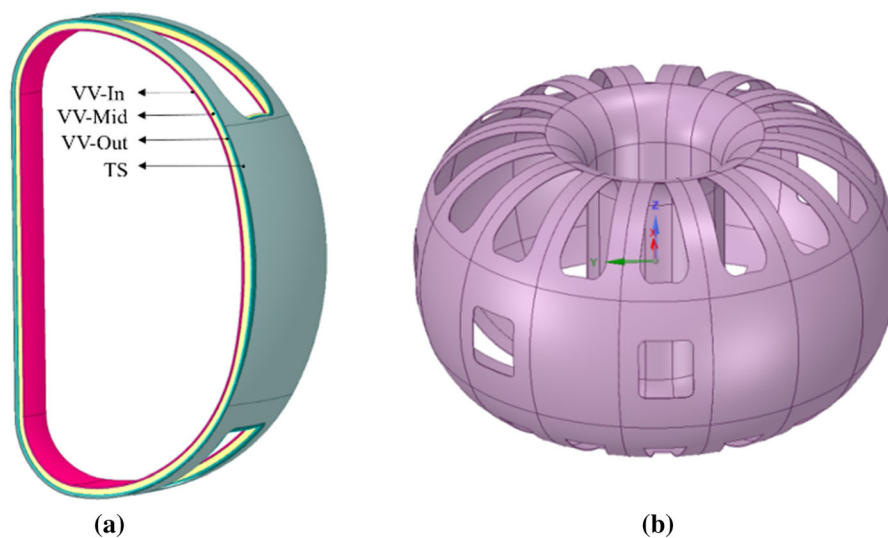
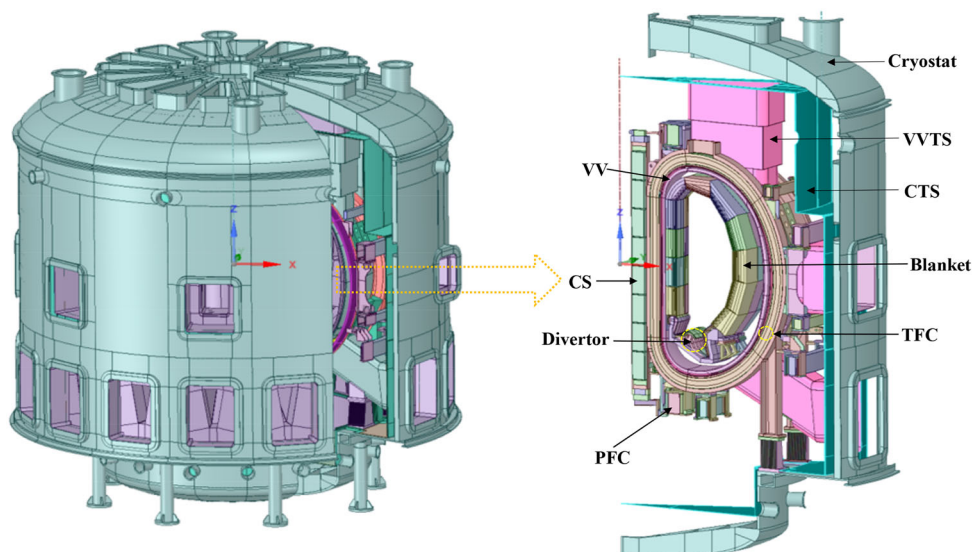


Fig. 3 (Color online) CAD engineering model of the CFETR



2 3D-shape segmentation algorithm

3D-shape segmentation technology has remained a research hotspot in the field of computer graphics. It has been widely used in model retrieval, feature recognition, and model reconstruction. Traditional segmentation methods, such as a conditional random field [23] and extreme learning machine [24], require manual definition of 3D-shape feature descriptors. The quality of feature extraction directly affects segmentation quality and directly depends on the quality of manually defined feature descriptors. In recent years, Convolutional Neural Networks (CNNs) have demonstrated excellent performance (recognition, classification, and object detection) in image classification, semantic segmentation, target detection, and other tasks. Compared to a traditional neural network, a CNN adopts a convolution kernel to extract features automatically. Generally, CNN concepts are useful for 3D-shape segmentation algorithms. The intrinsic data structure of a 3D-shape representation file does not resemble the regular and ordered datasets of images required by the CNNs. This leads to difficulties in the direct convolution of these 3D-shape representations. In light of the inadaptability of CNNs to the irregular structures of datasets, tricky strategies have been adopted to avoid convolution. For example, 3D shapes are typically converted into regular 3D voxel grids [18] or sets of images [19] (e.g., multi-views). Although volumetric representations have the advantage of simplicity, they are time consuming and require significant computational resources.

Point clouds are the simplest representation of 3D-shaped data because a point-cloud-based technique is superior in terms of easily accessible data acquisition and simple representation. Additionally, it is convenient to convert them into other representations. Therefore, CNNs for point clouds have been proposed in recent years. PointNet [20] is a representative application of a CNN for irregular and sparse point clouds and adopts a T-Net to obtain a transformation matrix that is used to initialize the point cloud and ensure the invariance of rotation and translation. To make the model invariant to input permutation, a symmetric function was designed to solve the disorder of the point cloud. Although PointNet is efficient, Multi-Layer Perceptron (MLP) is performed on a single point, ignoring local information, and its performance is poor for 3D shape segmentation. Furthermore, the density of cloud data was not considered in PointNet. Inspired by CNNs, PointNet++ [25] was proposed to partition points into local regions, extract local features at multiscale and multiresolution, and obtain deep features using a multi-layer network structure. PointCNN [26] is a generalized CNN capable of handling irregular data represented in

point clouds. The core of PointCNN is the X-Conv operator, which weighs and permutes input points and features.

Mesh is another representation used to draw and store 3D models in computer graphics. The primary advantage of mesh representation is that a significant amount of information can be easily stored in less memory. For example, a large but simple surface can be represented by a small number of polygons. Another advantage of a mesh is its connectivity. However, the intrinsic data structure of the mesh representation is complex and irregular. For example, the number of elements in a mesh may vary significantly between different 3D shapes, and their permutations are arbitrary. Despite these problems, meshes are more capable of describing 3D shapes than are other representation methods. In this case, effectively using a mesh to represent 3D graphics is a new challenge and related studies are limited. Recently, MeshCNN [17], which is specifically designed for triangular meshes, has been invariant to translation, rotation, and scaling. Simultaneously, convolution and pooling operations for irregular and non-uniform structures are redefined to allow the direct application of MeshCNN to irregular triangular meshes with better robustness and higher accuracy. Figure 4 shows the application of state-of-the-art deep learning networks to the segmentation of 3D models (the data in Fig. 4a are derived from [27] and the data in Fig. 4b are derived from [17]). PointNet, PointNet++, PointCNN, and MeshCNN exhibited relatively high classification accuracies. Considering segmentation, MeshCNN had a higher accuracy. Considering that CAD models are easily represented by a mesh, this study adopted MeshCNN as the deep learning framework.

MeshCNN first directly applies a CNN to a polygonal mesh by utilizing a similar definition and implementation of CNN standard blocks for irregular and non-uniform meshes. On the one hand, the basic object of convolution is an edge that is similar to a pixel in an image. As each edge is incident on at most two faces (triangles), a local and fixed-size neighborhood is defined in a simple manner for convolution on irregular structures. On the other hand, the designed features are invariant to rotation, scaling, and translation.

MeshCNN is applied to manifold meshes, which guarantees that each edge is connected to two triangular patches at most. This results in each edge being adjacent to the other four. As shown in Fig. 5, the neighbors of each edge e are edge a , edge b , edge c , and edge d . The vertices of a triangular patch are ordered counterclockwise and two orders are defined for the four neighbors of each edge. For example, the neighbors of edge e can be sorted into (a, b, c, d) or (c, d, a, b) . This blurs the convolutional receptive field and hinders the formation of the invariant features. Two

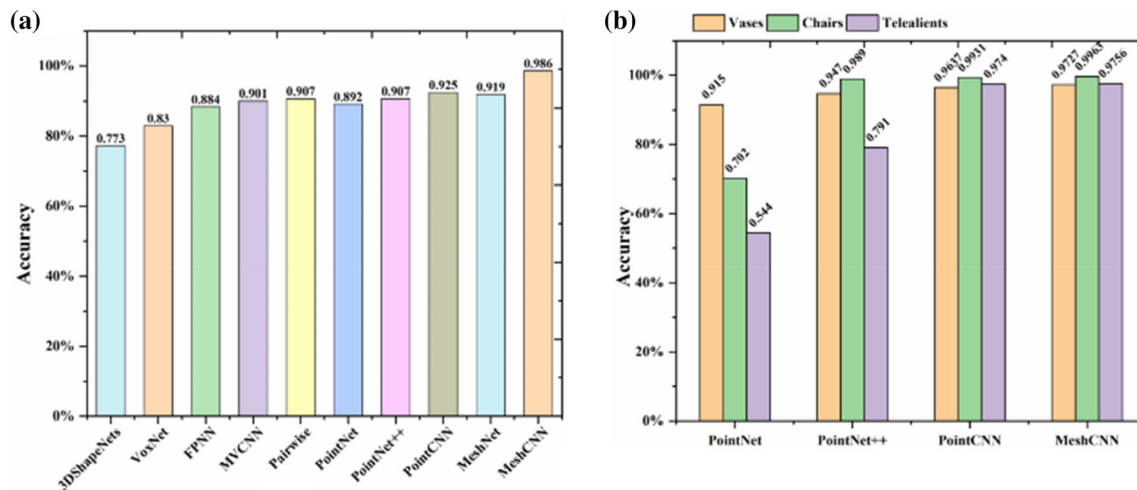
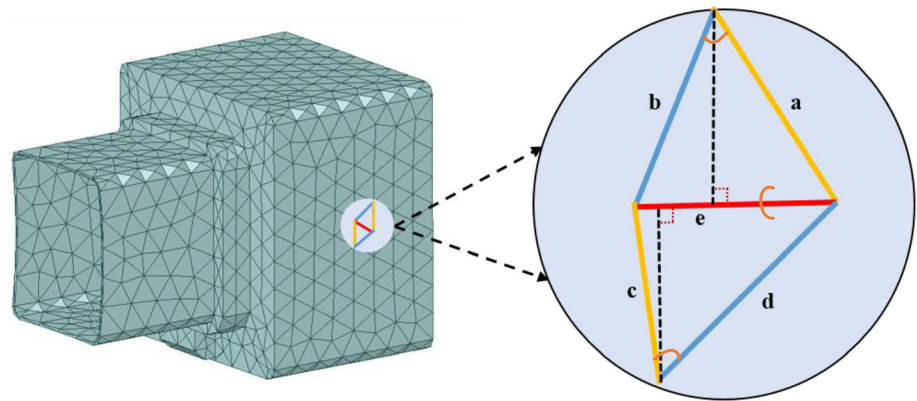


Fig. 4 (Color online) Outline of the converting methods for 3D shapes. The X-axis indicates the proposed method, and the Y-axis indicates the classification accuracy

Fig. 5 (Color online) A convolution structure for edge irregular triangular meshes



actions address this issue and guarantee invariance to the similarity transformations (rotation, translation, and scale).

MeshCNN adopts two methods to solve this problem, making the network convolution invariant. The main results are as follows.

- With respect to edge feature extraction, the features of each edge were carefully designed as a five-dimensional vector composed of the dihedral angle, two inner angles, and two edge-length ratios for each triangular patch. These features contain only relative geometric properties without absolute position information, and have invariance in rotation, scaling, and translation. Therefore, unlike common representations (such as point-based representations), features do not contain vertex coordinate information, which makes it better to generalize shape features and promote the invariance of the similarity transformation.
- Symmetric functions are designed to eliminate the fuzziness caused by the order of the edges, and the unique symmetry of the triangular mesh is used to

eliminate the duality of the neighborhood order to make it invariant to translation, rotation, and scaling. The four edges are aggregated into two pairs of edges (a and c , b , and d) and a sum (b and d), and new features are generated by applying a simple symmetric function to each pair ($abs(b, d)$ and $sum(b, d)$), which are then used for convolution.

3 MeshCNN-based brep to CSG conversion

3.1 Dataset generation for 3D CAD Model

Training datasets are crucial for deep learning. However, unlike images, it is difficult to create a dataset for 3D CAD models for four reasons. (1) The 3D CAD model is difficult to obtain and needs to be created manually, resulting in scarce availability of geometric datasets. (2) The more complex CAD model, which is generally represented by BREP, is unsuitable for most deep learning

methods. It is difficult to convert CAD models into regular-format data. (3) Labels must be manually marked for the 3D-shape segmentation, and manual labeling cannot be extended. (4) In many cases, the semantics of the CAD model are unclear, and it is difficult to represent a 3D CAD model using semantic parts. Therefore, this study focused on how to easily generate datasets. The following section presents a dataset creation method for the CAD models commonly used in fusion reactors.

3.1.1 Labeling

A geometric processing pipeline was developed based on the OCC CAD engine, which was used to deal with CAD models to generate mesh, edge label, and weight files. Triangulation of the CAD model is a key step in the design of a model dataset. OCC provides a powerful mesh-generation algorithm that supports multiple CAD file formats (such as STEP, IGES, and STL) and includes various integrated mesh-generation algorithms. Defining a CAD model as having different semantic parts is difficult because semantics depend on the preferences of the modeler. Figure 6 shows an example of a labeled CAD model. Figure 6a is a representative CAD model in the dataset, defined as S . Figure 6b is the ideal result of segmentation according to the preferences of the modeler, defined as S_{part} , which is an abbreviation of semantic parts (marked as Seg1–6). Figure 6c shows the face list for each semantic part. Figure 6d shows a special semantic part in which faces are not used to guide the construction of auxiliary splitting surfaces.

The detailed labeling flow is shown in Fig. 7.

- (a) Import a solid model S , traverse each face in S , and obtain an ordered list of faces defined as $Face_List$.

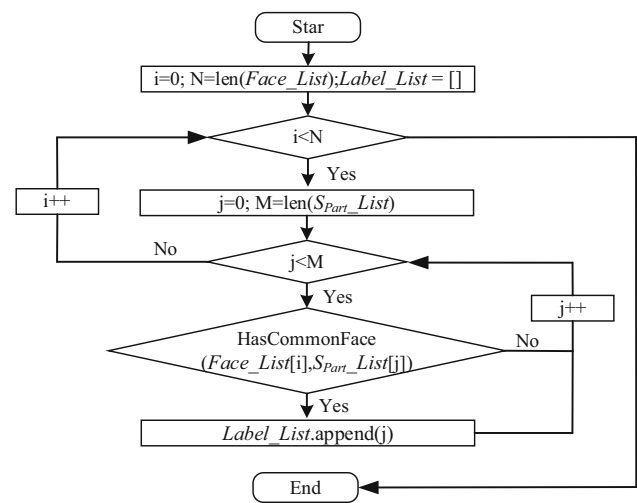


Fig. 7 Flowchart of model labeling

- (b) Import the subbody model S_{part} , traverse each solid in S_{part} , and obtain an ordered solid list, defined as $S_{\text{part_List}}$.
- (c) If $Face_list[i]$ is the same as the face in $S_{\text{part_list}}[k]$, the label of $Face_list[i]$ is set to k . If one face simultaneously belongs to more than one solid, the surface is marked as a special class, as shown in Fig. 6d.
- (d) Finally, the label of each face in solid S is obtained and defined as $Label_List$.

3.1.2 Generation of a training dataset

The proposed algorithm was integrated into cosVMPT, which is widely applied in the fusion field. Therefore, typical fusion CAD engineering models were selected to generate a training dataset. A training dataset (Fig. 8) consisting of five typical types of fusion device models was

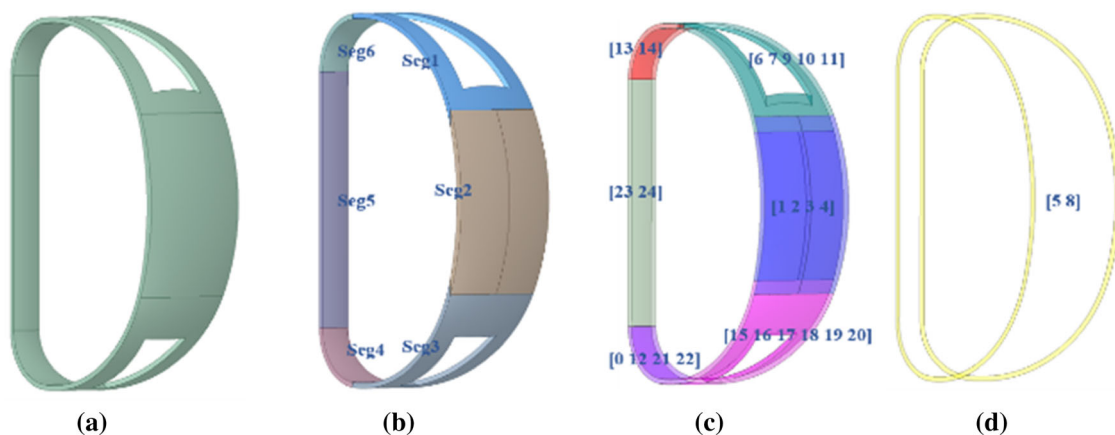


Fig. 6 (Color online) A diagram of labeling. **a** An example of a CAD model being labeled. **b** The ideal segmentation result. **c** Face list of each semantic part. **d** A special semantic part



Fig. 8 (Color online) Typical models in the training dataset

constructed; some of the models are shown in Fig. 8. These models include the cryostat, TFC, VV, and TS. The generation process of the dataset is as follows:

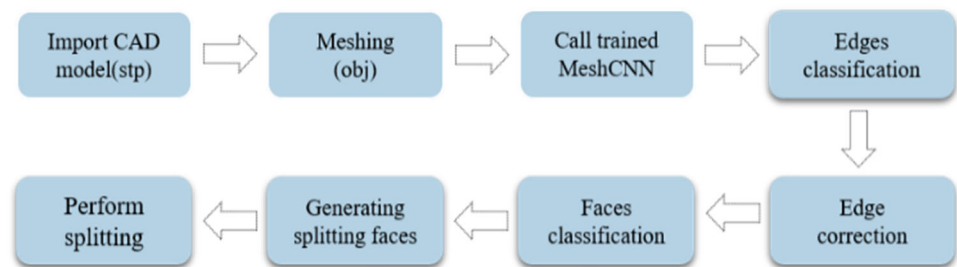
- (a) *Model labeling* Each model in Fig. 8 was labeled, as described in the previous section, and a label dictionary was obtained.
- (b) *Data augmentation* The number of models was increased through rotation, scaling, and translation. The name of the generated model is the same as that of the basic model, and it is numbered successively.
- (c) *Triangular meshing*.
- (d) *Triangular patch labeling* According to the model name (key), a label list for each face was obtained from the label dictionary and each triangular patch

was labeled according to its affiliation with each face.

- (e) *Generating Training Files* Mesh, edge label, and weight files were generated as training data for the MeshCNN.

3.2 Intelligent segmentation

This study also focused on a segmentation algorithm to achieve intelligent segmentation of a CAD model. The segmentation of a complete model requires the following processes (Fig. 9): (1) Import the CAD model to be converted. (2) Generate the corresponding mesh file. (3) Call-trained neural network for classification. (4) Export edge classification results of MeshCNN. (5) Modify the

Fig. 9 Flowchart of a complete pre-segmentation

classification results. (6) Obtain the classification results of the faces according to the affiliation relationship between the faces and edges and then group the faces according to the labels. (7) Generate auxiliary splitting surfaces based on the grouping results of the surfaces. (8) Segment the model using the generated splitting surfaces.

3.2.1 Edge correction

In many cases, some edges may be misclassified, as shown in Fig. 10a, which makes it difficult to generate split faces. However, the CAD models are rich in geometric information. Therefore, a strategy for edge correction was introduced to further improve the accuracy of edge classification. For each face, the category with the largest number was considered as the category of the face by counting the number of edges N_i of different types on each surface. Figure 10b shows the revised results.

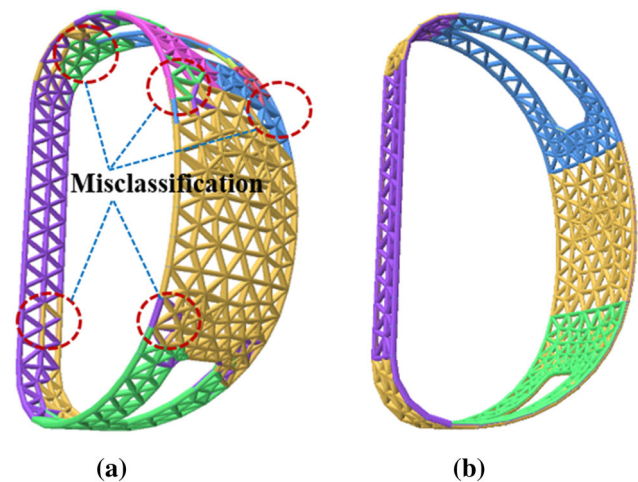
$$P_i = \frac{N_i}{\sum_{i=1}^n N_i} \quad (1)$$

3.2.2 Construction of common edge sets

The auxiliary splitting surface is constructed based on the edges of the model itself, and the classification result of each edge in the triangular patches cannot be directly used to generate an auxiliary splitting surface. Therefore, the first step in constructing an auxiliary splitting surface is to build common edges between the different segments (defined as *Seg_List*). The classification of each edge was obtained using MeshCNN, and the classification of each face was based on the affiliation relationship between faces and edges. Finally, faces with the same label were divided into segments, and a common edge set was generated by determining whether there were common edges between the segments. The implementation process is illustrated in Fig. 11.

3.2.3 Construction of auxiliary splitting faces

Considering the different types of shared edges (e.g., lines and arcs), the methods for generating auxiliary

**Fig. 10** (Color online) Classification results of edges using MeshCNN. **a** Classification results before correction. **b** Classification results after correction

splitting faces also differ. It is necessary to limit the size of the splitting surface to avoid excessive segmentation of the model caused by the excessive size of the splitting surface, which can be achieved by limiting the endpoints of the common edges. According to the type of common edge, the following situations are considered in Fig. 12: (1) When the shared edges are an arc, a plane is constructed with the arc. (2) When the shared edges are two arcs, a cylindrical surface is constructed if the generatrix is parallel to the axis. A conical surface is constructed if the generatrix intersects the axis. (3) When a shared edge is a straight line, a plane is constructed using the straight line and the axis of its connected curved surface. (4) When the shared edges are two straight lines, two planes are constructed using the two straight lines and axis of the connected curved surface. (5) For other curves, a plane is constructed for each curve.

4 Verification and applications

4.1 Performance evaluation

To verify the feasibility and efficiency of the intelligent pre-segmentation algorithm, certain components (Fig. 13)

Fig. 11 Flowchart of constructing common edge sets

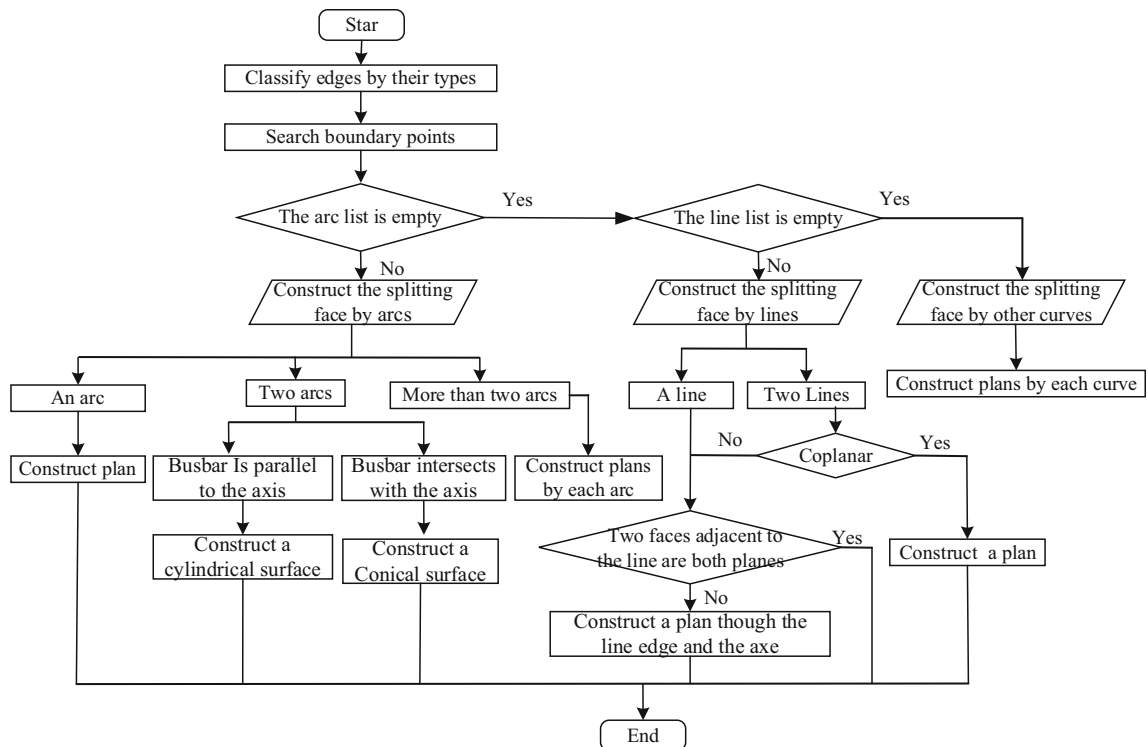
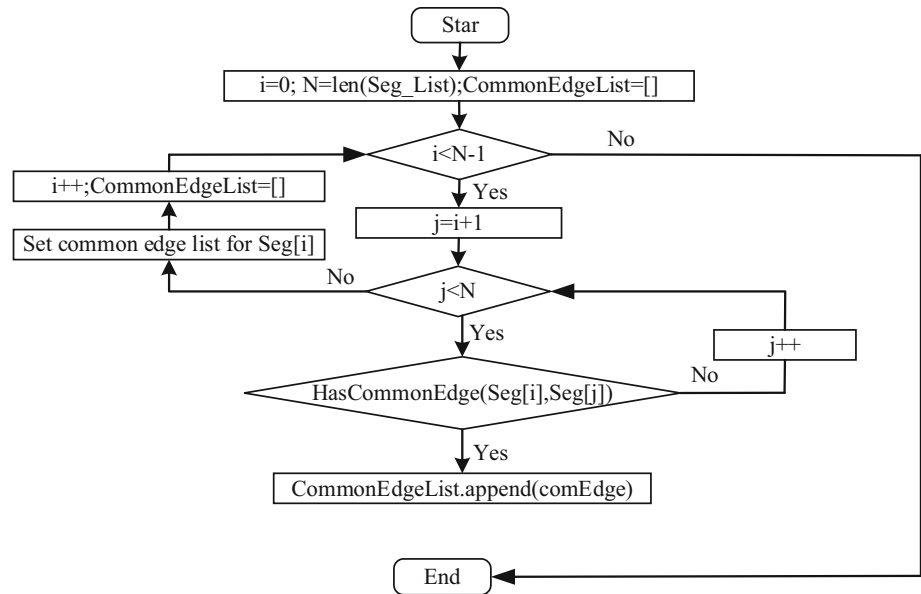


Fig. 12 Flowchart of the construction of auxiliary splitting faces

of the latest 360-degree CFETR engineering model were employed. The main feature of these components is that they are composed of a large number of high-order surfaces (torus), which often require manual preprocessing before they can be correctly converted using cosVMPTs. Figure 14 shows the decomposition results of the traditional

optimization algorithms and intelligent segmentation algorithms of a sector (22.5°) for the cryostat, TFC, VV, and TS. Figure 14d shows the decomposition results from cosVMPT for the intelligent pre-segmentation results and Fig. 14e shows the decomposition results of the original optimization algorithm. An outstanding advantage of

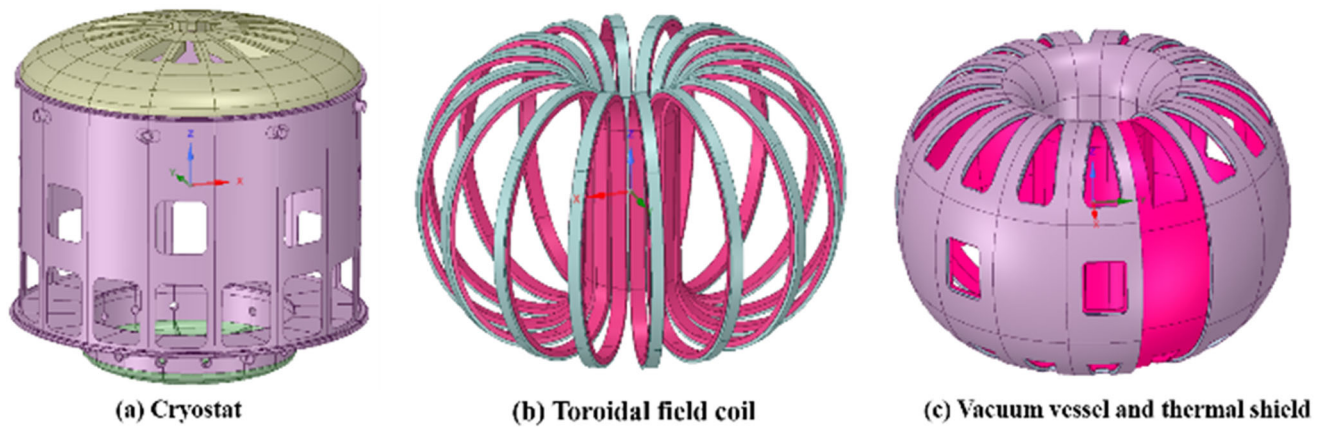


Fig. 13 (Color online) Some components of the latest 360-degree CFETR

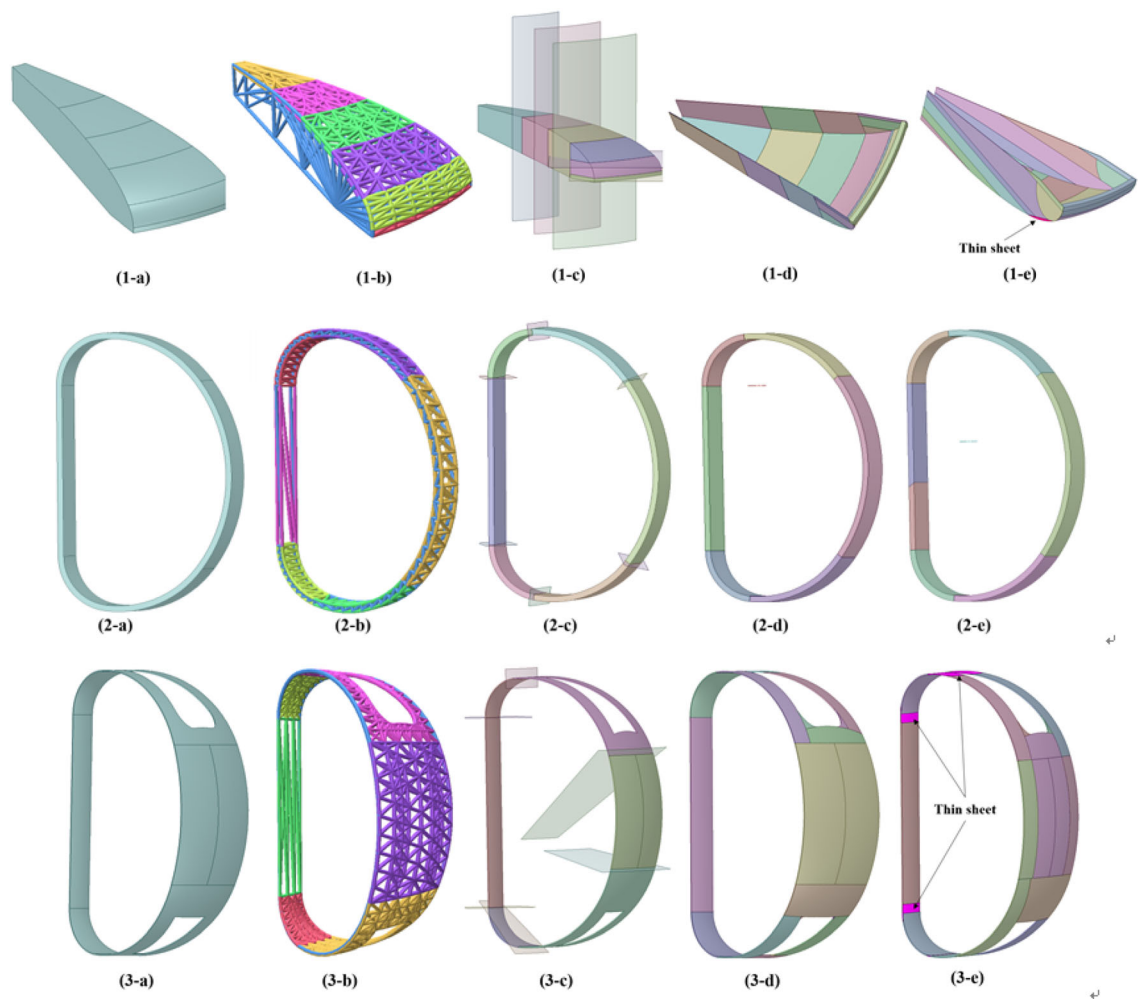


Fig. 14 (Color online) The decomposition results of traditional optimization and intelligent segmentation algorithms. **a** is the whole model to be converted, **b** is the classification result of triangular patches, **c** is the preliminary decomposition results of intelligent pre-

segmentation using generated auxiliary splitting faces, **d** is the final decomposition results from cosVMPT for the pre-segmentation results, and **e** is the decomposition results of the traditional optimization algorithm

Table 1 Comparison of the final convex solids

Components	Original algorithm	Traditional optimization algorithm	Intelligent pre-segmentation algorithm
Cryostat	*	1924	1676
TFC	413	224	193
VV and TS	*	1043	855

*Indicates models that cannot be converted correctly

Table 2 Comparison of the conversion CPU time

Components	Original algorithm	Traditional optimization algorithm (s)	Intelligent pre-segmentation algorithm (s)
Cryostat	*	3191	1034
TFC	138 s	87	68
VV and TS	*	6921	2583

*Indicates models that cannot be converted correctly

intelligent preprocessing is that the decomposed model is regular. Intelligent preprocessing prevents the generation of many thin sheets, which is the main reason for the failure of the OCC Boolean operation.

The quality of a conversion algorithm can be evaluated based on the number of final convex solids and the conversion CPU time. The more regular a model, the fewer the number of convex bodies, and the better the conversion algorithm. The presented algorithm was compared with other traditional algorithms based on cosVMPT [22]. Tables 1 and 2 compare the final convex solids and conversion CPU times of the original conversion algorithm without optimization, the traditional rule-based optimization algorithm, and our deep learning-based intelligent pre-segmentation algorithm. The results show that the intelligent pre-segmentation algorithm exhibits a significant improvement in performance and efficiency, particularly in the conversion time. For VV and TS, which consist of a large number of toruses, the CPU conversion time was reduced by twice to thrice. Simultaneously, according to the evaluation of the experienced modeler, the conversion result was more intuitive and consistent with the intention of the modeler.

In many cases, tangential solids may be generated during the conversion. Because of accuracy, when a model is complex, it often leads to the loss of particles in tangent places. The modeler must pre-segment the model manually to avoid the generation of tangent solids, which is time-consuming and tedious. Therefore, to reduce the generation of tangent solids, model datasets for local features, which often need to be handled manually, have been designed (e.g., VV). Figure 15 presents some of the results. It can be

observed that the segmentation result is more consistent with the modeler's intention, and avoids the generation of tangent solids.

4.2 Robustness evaluation

To further verify the robustness of the intelligent pre-segmentation algorithm, test cases were designed, such as locally modifying the model, transforming the model (translation, rotation, and scaling), and using models with similar structures but different parameters. Figure 16 shows the validation results, illustrating that the intelligent pre-segmentation algorithm exhibited strong robustness and invariance in terms of rotation, translation, and scaling. The partial absence of a model does not affect the segmentation accuracy. When the model structure was significantly changed, as shown in Fig. 16f, the classification results were not ideal. On the one hand, triangular patches in a symmetrical position have the same attributes, the semantics are not obvious, and they are not easy to distinguish. On the other hand, a face may be recognized as two half faces by the OCC; therefore, the triangular patches will be different. This can be further improved through model repair techniques (e.g., merging surfaces) and by increasing the diversity of the training sets. In addition, the auxiliary surface-generation program can be further improved.

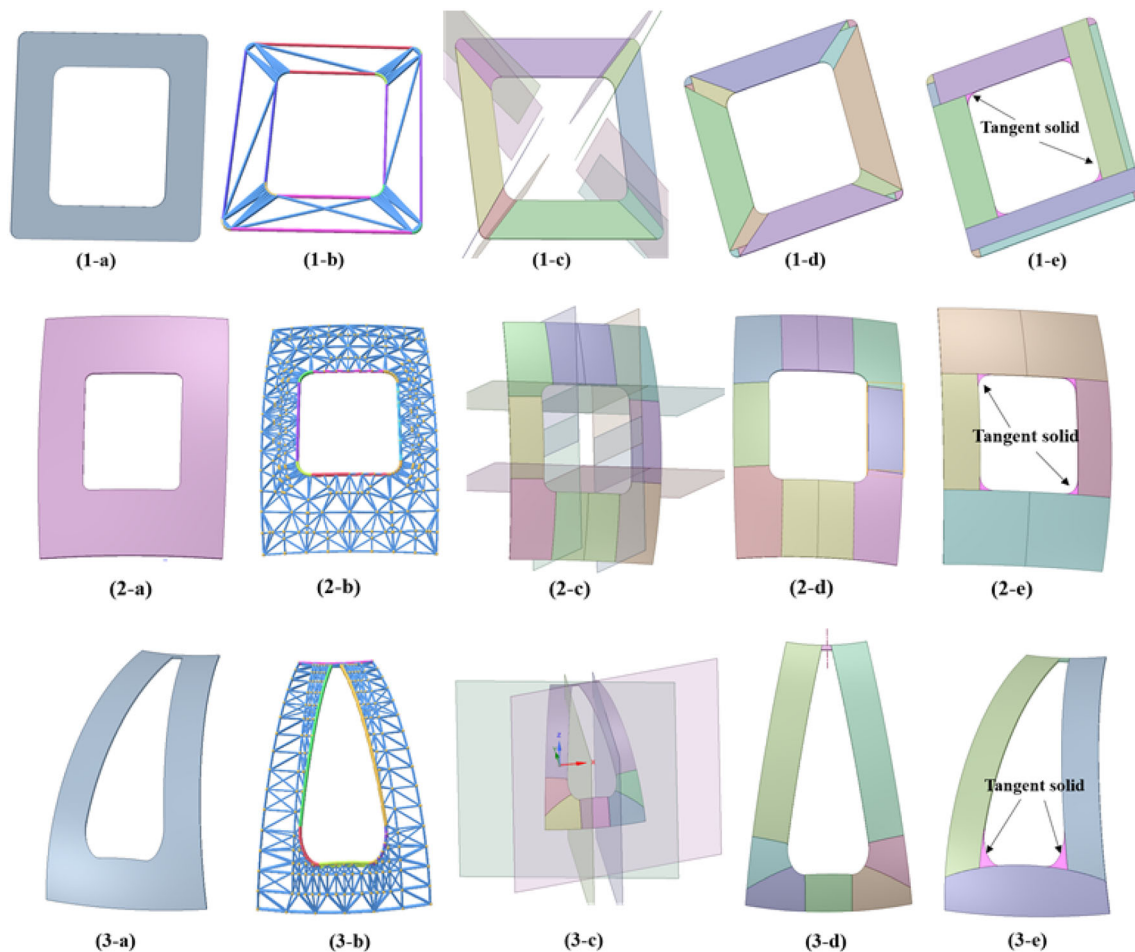


Fig. 15 (Color online) The decomposition results of traditional optimization and intelligent segmentation algorithms. **a** is the model part to be converted, **b** is the classification result of the triangular patches, **c** is the preliminary decomposition results of intelligent pre-

segmentation using the generated auxiliary splitting faces, **d** is the final decomposition results from cosVMPT for the pre-segmentation results, and **e** is the decomposition results of the traditional optimization algorithm

5 Conclusion and prospects

Separation-based BREP-CSG conversion has the advantages of being efficient, reliable, and easy to implement. It is widely used in computer-aided modeling of MC codes. However, separation-based BREP-CSG conversion relies heavily on a large number of Boolean operations, which may fail in certain complex models. To improve the conversion efficiency and avoid errors, a 3D-shape segmentation algorithm based on MeshCNN was creatively applied to optimize the BREP-CSG conversion while avoiding manual intervention in the model preprocessing. The main contributions of this study can be summarized as follows. (1) A general labeling method for CAD models and a fast-training dataset generation method were implemented based on OCC. (2) Considering the CAD engineering models commonly used in fusion devices, five types of datasets were designed. (3) Intelligent pre-segmentation was validated using the representative

components used in the CFETR. The results demonstrate that the intelligent pre-segmentation algorithm exhibits a significant improvement in performance and efficiency. The conversion CPU times of the VV and TS were reduced by two to three times. Moreover, the conversion result is more intuitive and consistent with the intention of the modeler, and avoids the generation of tangent solids. (4) The robustness of the intelligent pre-segmentation algorithm was verified. Clearly, the intelligent pre-segmentation algorithm exhibited strong robustness and invariance in terms of rotation, translation, and scaling.

Training models are very important for the proposed algorithm, and it is not currently easy to prepare a large training model. Therefore, it is necessary to develop a visual interactive model labeling method in which a model can be marked according to the intention of the modeler to clarify semantic information (such as triangular patches in a symmetrical position) in future. Because the proposed algorithm is a learning-based method, it is suitable for



Fig. 16 (Color online) Robustness test of the intelligent segmentation algorithms. **a** Translation. **b** Rotation. **c** Scaling. **d** Filling. **e** Holing. **f** With a similar structure but with different parameters

areas with multiple components of similar shape, such as the fusion reactor and reactor core, which are major application areas of related software (such as cosVMPT, MCAM, and McCad).

Author contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Yue-Tong Luo, Hua Du, and Yi-Man Yan. The first draft of the manuscript was written by Hua Du, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

1. K. Tang, T. Woo, Algorithmic aspects of alternating sum of volumes. Part 2: nonvergence and its remedy. *Comput. Aided Des.* **23**, 435–443 (1991). [https://doi.org/10.1016/0010-4485\(91\)90011-K](https://doi.org/10.1016/0010-4485(91)90011-K)
2. V. Shapiro, D.L. Vossler, Separation for boundary to CSG conversion. *ACM T. Graphic* **12**, 35–55 (1993). <https://doi.org/10.1145/169728.169723>
3. V. Shapiro, D.L. Vossler, Construction and optimization of CSG representations. *Comput. Aided Des.* **23**, 4–20 (1991). [https://doi.org/10.1016/0010-4485\(91\)90077-A](https://doi.org/10.1016/0010-4485(91)90077-A)
4. S.F. Buchele, R.H. Crawford, Three-dimensional half-space constructive solid geometry tree construction from implicit boundary representations. *Comput. Aided Des.* **36**, 1063–1073 (2004). <https://doi.org/10.1016/j.cad.2004.01.006>
5. B.L. Kirk, Overview of Monte Carlo radiation transport codes. *Radiat. Meas.* **45**, 1318–1322 (2010). <https://doi.org/10.1016/j.radmeas.2010.05.037>
6. Y.-T. Luo, *Research on Models Conversion and its Application in MCNP Modeling*. (Hefei University of Technology, 2005) (in Chinese).
7. L. Lu, Y. Li, A.-P. Ding et al., Application of MCAM in generating Monte Carlo model for ITER port limiter. *Chin. J. Nucl. Sci. Eng.* **27**, 3 (2007). <https://doi.org/10.3321/j.issn:0258-0918.2007.03.015>
8. Y. Li, L. Lu, A. Ding et al., Benchmarking of MCAM 4.0 with the ITER 3D model. *Fusion Eng. Des.* **82**, 15–24 (2007). <https://doi.org/10.1016/j.fusengdes.2007.02.022>
9. J. Xiong, G.-Z. Wang, D.-X. Wang et al., Applications of MCAM for ITER TRIPOLI 3D neutronics modeling. *Chin. J. Nucl. Sci. Eng.* **31**, 162–168 (2011)
10. F. Moro, U. Fischer, L. Lei et al., Applications of McCad for the automatic generation of MCNP 3D models in fusion neutronics. In: *Paper presented at 2013 IEEE 25th Symposium on Fusion Engineering* (San Francisco, CA, USA, 2013), pp. 10–14. <https://doi.org/10.1109/SOFE.2013.6635377>
11. F. Moro, U. Fischer, L. Lu et al., The McCad code for the automatic generation of MCNP 3-D models: applications in fusion neutronics. *IEEE Trans. Plasma Sci.* **42**, 1036–1041 (2014). <https://doi.org/10.1109/TPS.2014.2308957>
12. L. Lu, Y.F. Qiu, U. Fischer, Improved solid decomposition algorithms for the CAD-to-MC conversion tool McCad. *Fusion Eng. Des.* **124**, 1269–1272 (2017). <https://doi.org/10.1016/j.fusengdes.2017.02.040>
13. Y.-T. Luo, J. Sun, Y. Xue, Three-dimensional CSG tree construction from BREP model based on conversion feature. *Modern Electron. Technol.* **20** (2007). (in Chinese).

14. S.L. Liu, Y. Pu, X.M. Cheng et al., Conceptual design of a water cooled breeder blanket for CFETR. *Fusion Eng. Des.* **89**, 1380–1385 (2014). <https://doi.org/10.1016/j.fusengdes.2014.01.065>
15. S. Liu, X. Ma, K. Jiang et al., Conceptual design of the water cooled ceramic breeder blanket for CFETR based on pressurized water cooled reactor technology. *Fusion Eng. Des.* **124**, 865–870 (2017). <https://doi.org/10.1016/j.fusengdes.2017.02.065>
16. Y.T. Feng, Y.F. Feng, H.X. You et al., MeshNet: Mesh neural network for 3D shape representation. In: *Paper presented at 2019 Thirty-third AAAI Conference on Artificial Intelligence* (Hawaii, USA, 2019). <https://doi.org/10.1609/aaai.v33i01.33018279>
17. R. Hanocka, A. Hertz, N. Fish et al., MeshCNN: A network with an edge. *ACM. T. Graphics* **38**, 90 (2019). <https://doi.org/10.1145/3306346.3322959>
18. Z.R. Wu, S.R. Song, A. Khosla et al., 3D ShapeNets: A deep representation for volumetric shapes. In: *Paper presented at 2015 IEEE Conference on Computer Vision and Pattern Recognition* (USA, 2015). <https://doi.org/10.1109/CVPR.2015.7298801>
19. A. Angrish, A. Bharadwaj, B. Starly, MVCNN++: Computer-aided design model shape classification and retrieval using multi-view convolutional neural networks. *J. Comput. Inf. Sci. Eng.* **21**, 011001 (2021). <https://doi.org/10.1115/1.4047486>
20. R. Qi Charles, H. Su, M. Kaichun et al., PointNet: Deep learning on point sets for 3D classification and segmentation. In: *Paper presented at 2017 IEEE Conference on Computer Vision and Pattern Recognition* (Hawaii, USA, 2017). <https://doi.org/10.1109/CVPR.2017.16>
21. H. Du, Y.T. Luo, C.C. Han et al., Development of an assistant program for CAD-to-cosRMC modelling. *Fusion Eng. Des.* **157**, 111662 (2020). <https://doi.org/10.1016/j.fusengdes.2020.111662>
22. H. Du, Q.R. Wu, P. Lu et al., Development of cosVMPT and application of creating 3D neutronics model for 360-degree CFETR. *J. Fusion Energ.* **40**, 2 (2021). <https://doi.org/10.1007/s10894-021-00299-0>
23. E. Kalogerakis, A. Hertzmann, K. Singh, Learning 3D mesh segmentation and labeling. *ACM T. Graphic.* **29**, 102 (2010). <https://doi.org/10.1145/1778765.1778839>
24. Z.G. Xie, K. Xu, L.G. Liu et al., 3D shape segmentation and labeling via extreme learning machine. *Comput. Graph. Forum.* **33**, 5 (2014). <https://doi.org/10.1111/cgf.12434>
25. C.R. Qi, L. Yi, H. Su et al., PointNet++: deep hierarchical feature learning on point sets in a metric space. *Adv. Neur. Inf.* (2017). <https://doi.org/10.48550/arXiv.1706.02413>
26. Y.Y. Li, R. Bu, M.C. Sun et al., PointCNN: convolution on X-transformed points. *Adv. Neur. Inf.* (2018). <https://doi.org/10.48550/arXiv.1801.07791>
27. Y.T. Feng, Y.F. Feng, H.X. You et al., MeshNet: Mesh neural network for 3D shape representation. In: *Paper presented at 2018 AAAI 32nd Conference on Artificial Intelligence* (Louisiana, USA, 2018) <https://doi.org/10.1609/aaai.v33i01.33018279>