Advances in Engineering Software 86 (2015) 55–69

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

# Automatic surface repairing, defeaturing and meshing algorithms based on an extended B-rep

Jianjun Chen [a], Bingwan Cao [a], Yao Zheng [a], Lijun Xie [a,*], Chenfeng Li [b], Zhoufang Xiao [a]

[a] Center for Engineering and Scientific Computation, and School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China
[b] Zienkiewicz Centre for Computational Engineering, and Energy Safety Research Institute, Swansea University, Swansea SA2 8PP, UK

ARTICLE INFO

*Article history:*
Received 10 October 2014
Received in revised form 14 March 2015
Accepted 5 April 2015
Available online 29 April 2015

*Keywords:*
Mesh generation
Repairing
Defeaturing
Boundary representation
Virtual topology
Face clustering

ABSTRACT

This paper presents an extended surface boundary representation (B-rep), where each topology entity can have dual geometric representations to accommodate various defects (e.g., gaps and overlaps) commonly present in CAD models. Keeping a uniform B-rep and the unsuppressed geometry data enables the use of various existing repairing, defeaturing and meshing algorithms to process CAD models with small gaps and overlaps on surface boundaries. The continuous geometry of the input model remains untouched in the repairing, defeaturing and meshing process, and the output mesh is loyal to this geometry. Such feature is often desirable in numerical simulations that require meshes with high geometry fidelity.



## 1. Introduction

To facilitate a numerical simulation, one needs to first prepare a geometry model and then discretise this model into a mesh. These two steps together are called the pre-processing of the simulation. For a simulation with complex geometry configuration, both steps are major performance bottlenecks because of intensive manual interactions. Recent advances of automatic mesh generation technologies relieve the burden of mesh generation to some extent. However, in many cases, creating qualified geometry inputs for mesh generation remains an outstanding challenge [1,2]. For instance, it was reported that, in Sandia National Laboratory, geometry preparation and mesh generation account for about 60% and 20% of overall simulation time, respectively, while only 20% of the overall time is actually devoted to numerical simulation [3].

Most analysis models come from Computer Aided Design (CAD) systems. The original CAD models rarely meet the requirement of mesh generation and they need to be processed following a two-step procedure. Firstly, a CAD model is often *contaminated* by geometry errors, which if not rectified may collapse or invalidate the meshing procedure. Therefore, a *repairing* procedure is required to *clean* the CAD model and remove such geometry errors. Secondly, the input CAD model may contain many details for manufacturing or other purposes, but the meshing algorithm may fail to recognise them and generate badly shaped elements under larger mesh sizes than the geometry scales of the fine details. Even if the meshing algorithm captures these details and generates a mesh with acceptable quality, the total element number may increase massively due to small element sizes defined around the details. Therefore, a *defeaturing* step is indispensable to remove the details with minor analysis significance and ensure the output mesh can define a cost-effective analysis model.

Mesh generation starts from surface meshing. In most cases, the subsequent volume meshing [4] relies on the surface mesh only and does not access the CAD model. Therefore, to some extent, the most important step of geometry preparation is to define a clean and defeatured surface model. In the CAD and Computer Graphics (CG) communities, many methods have been proposed to depict a surface. However, the *continuous* representation based on NURBS, Bézier and Coons patches and the *discrete* representation based on triangular facets presently dominate in the community of Computer Aided Engineering (CAE). Accordingly, techniques for surface repairing and defeaturing can be categorised into two groups: those applied directly to the continuous model and those to the discrete model.

Geometry computations defined on a discrete model are very efficient because the edges and facets that depict the surface have linear equations. In contrast, a continuous surface has a high-order irreversible parametric representation. Many computations define

* Corresponding author.
*E-mail address:* zdxlj@zju.edu.cn (L. Xie).

http://dx.doi.org/10.1016/j.advengsoft.2015.04.004
0965-9978/

on it depend on numerically instable iterative procedures, hence more time-consuming and less reliable. Nevertheless, the continuous surface has its own strengths. Firstly, the design of most industry products is based on continuous surfaces. If the computational mesh is defined on a discrete model, the simulation accuracy may degrade due to the geometry inaccuracy of the mesh. Secondly, the continuous surface is usually organised with a boundary representation (B-rep). Many efficient defeaturing and meshing algorithms have been developed by utilising the topology entities of the B-rep [5–9]. These algorithms cannot be applied on a pure discrete model that is internally organised with some edge-based data structures.

In this study, we propose a new data structure namely the *hybrid surface B-rep*. It combines the continuous and discrete representations of surfaces. Each topology entity of the surface B-rep has a dual geometry representation in default. In addition, the mappings are maintained between the topology entities (*faces*, *curves* and *points*) of the B-rep and their counterparts (*facets*, *edges* and *vertices*) on the discrete model. The primary motivation of developing this data structure is to enrich our choices of geometry repairing, geometry defeaturing and mesh generation algorithms to serve different application purposes better. Upon this data structure, we can combine different types of algorithms in one pre-processing workflow. In contrast, most of the existing pre-processing approaches are solely based on either the discrete or the continuous model. As a result, they can only employ one type of geometry repairing, geometry defeaturing and mesh generation algorithms.

To demonstrate the benefit of the proposed data structure, a novel pre-processing workflow is proposed for CAD models with small gaps and overlaps on surface boundaries. These errors are common in CAD models due to translation errors, numerical inaccuracies and tolerance settings. For such CAD models, the existing pre-processing approaches prepare the mesh model by one of the following two approaches.

(1) *The continuous model based approach.* This approach is adopted by many commercial tools [10]. It first repairs the gaps of the input model directly by redefining the curves and surfaces in the neighbourhood [11]. No matter *real* [12] or *virtual* operations [5–7,13,14] are executed in the following defeaturing step, the final mesh is always generated on a modified CAD model.
(2) *The discrete model based approach.* This approach repairs the tessellated dual of the input CAD model [15–17] and if needed, the repaired discrete model can be further simplified [18]. All of these steps modify the geometry and take the risk of degrading the geometry accuracy of the final mesh.

The proposed approach combines the discrete model based geometry-repairing algorithm [15,16] and the continuous model based defeaturing algorithm [6] in one pre-processing workflow. Firstly, it repairs the discrete dual of the input model, and updates the topology entities of the B-rep. Next, the repaired B-rep is defeatured by combining neighbouring faces. The defeatured B-rep contains two types of faces: the *real* faces and the *virtual* faces. The *real* face has a unique counterpart in the input model and its continuous model is meshed directly by a mapping-based surface mesher. However, the *virtual* face, i.e. a combination of many faces, has no intrinsic parametric space. Its discrete model is meshed instead and then the interior mesh points are projected onto the continuous model. The final output is a combination of the meshes generated on real and virtual faces.

Compared with the algorithms solely based on the discrete or the continuous models, the advantages of the proposed approach are:

(1) Most geometry computations are performed on the discrete model, avoiding the tedious, expensive and unreliable calculations on the continuous model.
(2) The input continuous model is not modified, and the output mesh is loyal to the input model rather than a discrete one or a modified one due to the repairing and defeaturing algorithms. This feature is often much desirable in simulations that require meshes with high geometry fidelity.
(3) The repairing algorithm not only removes geometry errors on the discrete model, but it also generates a correct B-rep. Therefore, the defeaturing and meshing algorithms that rely on a valid B-rep input can be directly applied.

## 2. Related work

If the CAD input is a continuous surface model with local geometry errors, a basic idea to address these errors is to redefine the geometry and topology entities locally [10,11]. For instance, in the region where two boundary curves overlap with each other, a new curve can be defined to replace the two curves. Accordingly, the topology entities of these two curves in the B-rep need to be replaced as well so that the adjacency of the faces initially bounded by the replaced curves is correctly set up. The advantage of this strategy is that both the geometry and the topology of the repaired model are continuous, and are accurately redefined so that a routine implementation of the downstream defeaturing and meshing algorithms is facilitated. Nevertheless, this strategy complicates the implementation of the repairing algorithm itself. Different overlapping cases and gap patterns must be classified to manage them accordingly. Redefining existing curves or faces and creating new curves or faces involve time-consuming and numerically instable geometry computations. Meanwhile, the modification of the CAD model may degrade the geometry fidelity of the output mesh.

Alternatively, some algorithms have been proposed to repair geometry errors on the discrete model, where the repairing procedure is reduced to a combination of local operations such as vertex contraction, edge splitting and so on [15–17]. These computations are usually very efficient because of their local and linear nature. The drawbacks of these repairing algorithms are also evident, despite of their simplicity and efficiency. Firstly, the geometry accuracy of the output mesh may be unqualified for a high-fidelity simulation. Moreover, the discrete model contains no high-level topology entities such as those defined in a B-rep; therefore, many effective defeaturing and meshing algorithms cannot be applied after repairing.

Defeaturing algorithms have also been developed separately for the continuous and the discrete models. The defeaturing algorithms for discrete models utilise local operations such as edge-collapse and vertex-removal, where the mesh geometry and topology are altered simultaneously [18]. However, the defeaturing algorithms for continuous models can be either *geometry-based* or *topology-based*. The geometry-based algorithms employ complicated geometry computations to change the model geometry directly [10,11], while the topology-based algorithms only alter the topology entities of a B-rep, hence more efficient and reliable [5–7,13,14]. However, the standard B-rep must be extended by introducing new types of topology entities such as *quilt* [13], *virtual topologies* [5] and *Mesh Constraint Topologies (MCT)* [14]. The extended B-rep is incompatible with the current neutral CAD file standards; thus, how to transfer it between different systems remains an issue. Nevertheless, this issue is not prominent for an integrated system where all pre-processing steps of numerical simulations can be completed without exchanging intermediate data with other systems. Presently, some geometry processing engines developed for mesh generation (such as CAPRI [19,20] and CGM [21,22]) support these new concepts, so do a few other mesh generation tools (such as Cubit [23] and Pointwise [24]).

Hamri et al. [25] also incorporated a dual geometry representation in their conceptual software environment for CAD/CAE integration. However, unlike the present study that treats CAD inputs with various geometry errors, their system assumes the CAD model is clean and without geometry defects. In addition, their defeaturing algorithm relies on the discrete model, while the proposed approach is topology-based and defined on the extended B-rep.

Besides, Smith et al. [17] and Quadros et al. [18] have foreseen the possible benefits of maintaining a dual geometry representation when developing their repairing and defeaturing algorithms. However, no details were given on how to organise a dual representation and how to implement their algorithms based on the dual representation. The algorithms depicted in [17,18] are purely based on discrete models, where the input and output meshes are both defined on discrete models. Different from existing approaches, this study takes a continuous model as the input, and the output mesh is defined on the continuous model.

## 3. Surface B-rep and its extension

### 3.1. The B-rep of a surface

The surface B-rep is a subset of the solid B-rep and it includes three basic topology entities: *face*, *curve* and *point*, as illustrated in Fig. 1. Meanwhile, a specific topology entity named *loop* is used to limit the valid region of a face. Internally, a loop refers to a set of boundary curves and is a *group entity* that distinguishes with other topology entities.

### 3.2. Extension I: the dual geometric representation

#### 3.2.1. Basic concept

In the extended B-rep, each topology entity has two types of geometric representations in default. One representation corresponds to geometry entities defined on the continuous model, while the other representation describes a surface defined by a set of triangles and a curve defined by a set of linear segments. Fig. 2 presents a simple example to illustrate this dual representation.

Depending on the input model, the dual representation can be initialized in two ways. If the input geometry is a continuous model, its curves and faces are meshed into linear segments and triangles, respectively. At this stage, the triangle shapes do not matter, but the tessellated model must be geometrically close to the continuous model. If the input geometry is a discrete model, it is first subdivided into many patches, after which a continuous representation is built by parameterising each patch. The first approach is employed in this study because a continuous input is considered.

A discrete model is composed of many triangles, and it is represented by using the half-edge data structure, which contains three topology entities of different dimensions: *facets*, *edges* and *vertices*.

#### 3.2.2. The mappings between the B-rep and the discrete model

To connect the B-rep and the discrete model, three basic mappings between their topology entities are defined as follows:

(1) *The face-facet mapping.* A face corresponds to a set of facets.
(2) *The curve-edge mapping.* A curve corresponds to a set of edges.
(3) *The point-vertex mapping.* A point corresponds to a vertex.

Other mappings can be defined as well, e.g., between a curve and all vertices that lie on the curve, or between a face and all edges that bound the face. As these additional mappings can be derived from the basic mappings, they are not explicitly represented in the extended B-rep.

Two definitions are introduced below to describe the above mappings:

**Definition 1** (*Classification [26]*). Given a $d_i$-dimensional topology entity ($d_i = 0$–$2$) $M^{d_i}$ of the discrete model, $M^{d_i}$ is *classified* on a $d_j$-dimensional topology entity ($d_i \leqslant d_j \leqslant 2$) $G^{d_j}$ of the B-rep if $M^{d_i}$ lies on $G^{d_j}$, denoted as $M^{d_i} \sqsubseteq G^{d_j}$.

**Definition 2** (*Reverse Classification Set, RCS*). Given a $d$-dimensional topology entity ($d = 0$–$2$) $G^d$ of the B-rep, the $d$-dimensional topology entities of the discrete model classified on $G^d$ form a *reverse classification set*, denoted as $RCS(G^d) = \{M^d | M^d \sqsubseteq G^d\}$.

### 3.3. Extension II: virtual topology

In the proposed defeaturing algorithm, neighbouring faces need to be combined into a *virtual face*. Sheffer et al. [5] also proposed a similar but more generalised concept, *virtual topology*, which introduces more *virtual* operations to suppress features.

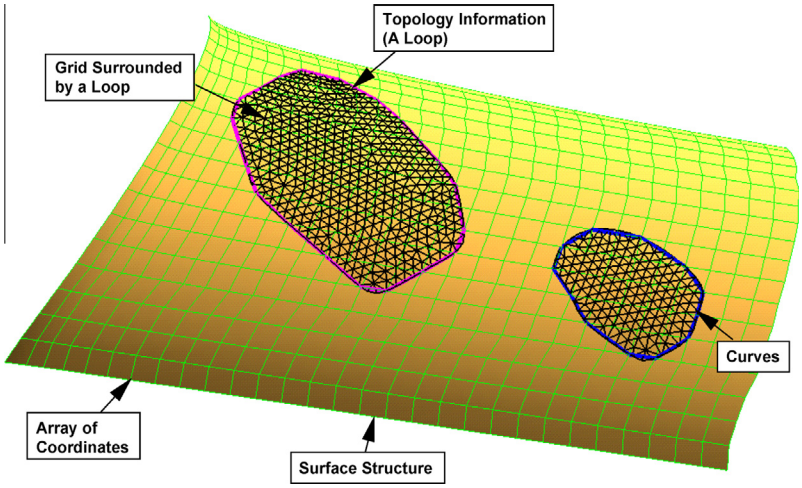


**Fig. 1.** Illustration for the surface B-rep.

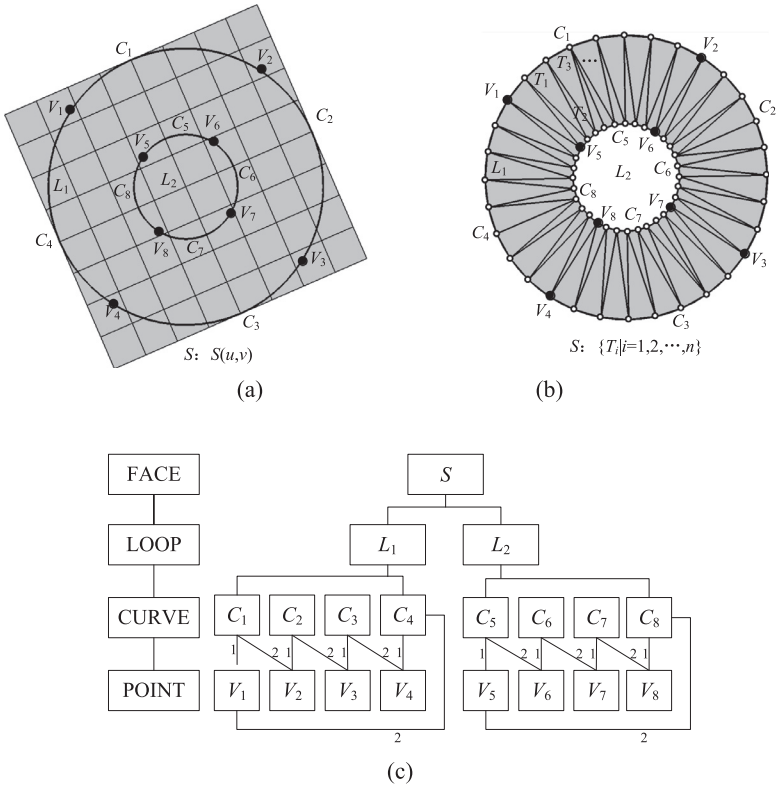


**Fig. 2.** The dual representation of a surface. (a) The continuous representation. (b) The discrete representation. (c) The B-rep.

Compared with tedious and expensive real operations that edit the geometry directly, their virtual counterparts only edit the B-rep topology entities. For instance, to combine two neighbouring faces $G_1^1$ and $G_2^1$ into a new face $G^1$, the real operation may create a new face through resampling to cover the original faces and then update the B-rep accordingly, while the virtual operation only creates a new topological face $G^1$ in the B-rep referring to $G_1^1$ and $G_2^1$. Assuming that $L_1$ and $L_2$ are the loops of $G_1^1$ and $G_2^1$, the loop of $G^1$ is $L = (L_1 \cup L_2) - (L_1 \cap L_2)$.

## 4. Overview of the method

The input is a composite surface model with each surface defined by a continuous representation. Small gaps and overlaps may be present on surface boundaries. One choice to obtain a qualified surface mesh is to first build a discrete representation, then repair the discrete model, and finally generate the mesh on the discrete model. However, to ensure a higher accuracy it is desirable to output the surface mesh directly defined on the original continuous model. Therefore, as shown in Fig. 3, a new workflow is proposed following the extended B-rep:

(1) *Initialisation.* The input model is tessellated to set up a dual geometric representation.
(2) *Repairing.* The discrete model in the dual representation is first repaired by executing iteratively vertex pair contraction and expansion operations [15,16]. Then, based on the mapping with the repaired discrete model, the topology entities of the B-rep are repaired to obtain a *clean* B-rep.
(3) *Defeaturing.* Neighbouring faces in the clean B-rep are combined to improve the *meshability* of the model. Consequently, the simplified B-rep contains *virtual faces*.
(4) *Meshing.* A *real face* corresponds to a face in the continuous input model, and can be meshed by a mapping-based surface mesher [27]. A *virtual face* refers to a combination of faces without intrinsic parameterisation, and its discrete model is parameterised and meshed, after which the mesh points are projected onto the continuous model. Finally, the meshes on real and virtual faces are merged for output.

## 5. Surface repairing

Geometry errors in the continuous input model are inherited by the discrete model after tessellation. The algorithm proposed by Patel et al. [15,16] is employed to produce a valid discrete model. For the sake of completeness, the operation is briefly recapped in Section 5.1. A novel algorithm is proposed to repair the B-rep, which is essential for the subsequent defeaturing and meshing algorithms, and this is explained in detail in Section 5.2.

### 5.1. Repairing the discrete model

The discrete model is repaired through the following steps [15,16]:

(1) Label the edges adjacent to only one facet as *boundary edges* and the ending vertices of these edges as *boundary vertices*.
(2) For a given resolution tolerance $\varepsilon_r$, search all pairs of boundary vertices $< P_i, P_j >$ that meet the condition $|P_iP_j| \leqslant \varepsilon_r$, and sort the pairs in an ascending order of $|P_iP_j|$.
(3) For each pair of boundary vertices $< P_i, P_j >$, if $|P_iP_j| \leqslant \varepsilon_g$, where $\varepsilon_g$ denotes a given a glue tolerance, contract them by *stitching*; otherwise, if $\varepsilon_g < |P_iP_j| \leqslant \varepsilon_r$, cover the gap between them by *filling*.

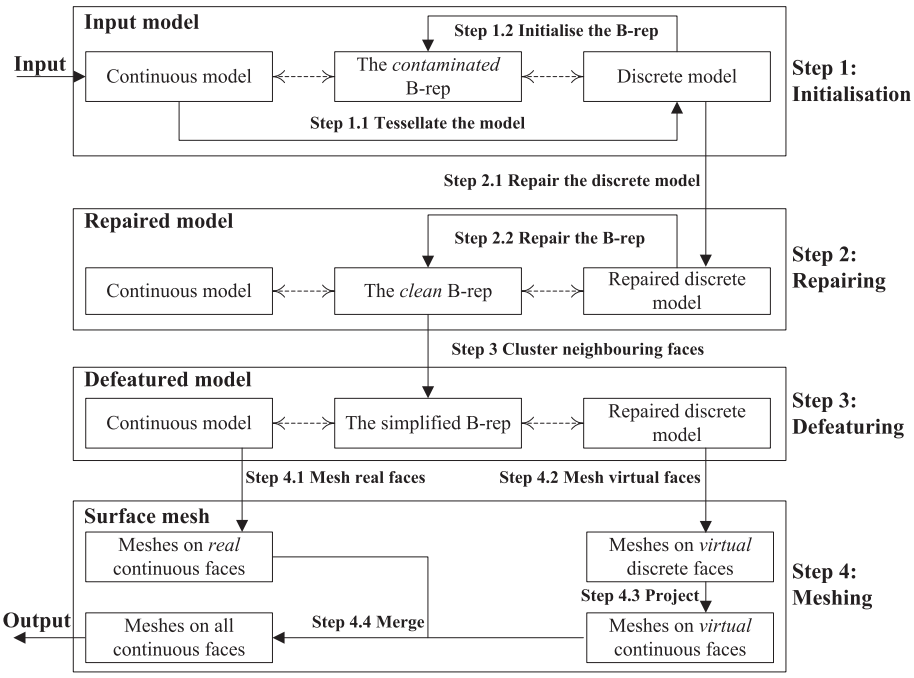


**Fig. 3.** General workflow of the proposed approach.

Besides the stitching and filling operations, an *edge splitting* operation is defined to deal with T-connections, as shown in Fig. 4, where $|P_1P_4| \geqslant \varepsilon_r$ and $|P_1P_5| \geqslant \varepsilon_r$, but the distance between $P_1$ and $P_4P_5$ is smaller than $\varepsilon_r$. Edge splitting introduces the projection of $P_1$ on $P_4P_5$, i.e., $P_7$. Stitching or filling can then be executed on the vertex pair $< P_1, P_7 >$.

Fig. 5 illustrates three cases of stitching. In Fig. 5a, $P_1$ and $P_2$ are contracted because $|P_1P_2| \leqslant \varepsilon_g$. Stitching may introduce edge contraction, as shown in Fig. 5b, where $P_1P_3$ and $P_2P_3$ are contracted when $P_1$ and $P_2$ are contracted. In the case of a T-connection, edge splitting is executed before stitching, as shown in Fig. 5c.

Fig. 6 illustrates two cases of filling. In Fig. 6a, two triangles are created to fill the gap between $P_1$ and $P_2$ because $|P_1P_2| > \varepsilon_g$. It is unsuitable to execute stitching for this vertex pair in order to maintain the geometry shape. Moreover, in the case of a T-connection, edge splitting must be executed before filling, as shown in Fig. 6b.

The efficiency of the algorithm largely depends on the search of boundary vertex pairs. The octree data structure can be adopted to speed up this searching procedure. Furthermore, an adaptive strategy for the settings of $\varepsilon_r$ and $\varepsilon_g$ can improve the reliability of the algorithm. More details are referred to [15,16].

### 5.2. Repairing the B-rep

The mapping defined in Section 3.2.2 connects the B-rep and the discrete model, and must be updated simultaneously when repairing the discrete model. The updated mapping is used to setup a correct B-rep based on the repaired discrete model.

#### 5.2.1. Update the mapping

Figs. 4–6 present all cases of the edge splitting, stitching and filling operations adopted by the repairing algorithm for the discrete model. The mapping between the B-rep and the discrete model is updated accordingly in each case of these operations.

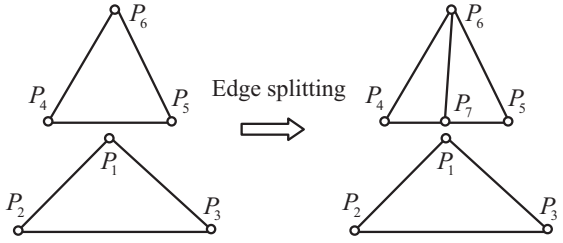


**Fig. 4.** Edge splitting.

(1) As shown in Fig. 4, edge splitting introduces a new vertex $P_7$ to split $P_4P_5$ and $\Delta P_4P_5P_6$. If $P_4P_5 \sqsubseteq G_i^1$ and $\Delta P_4P_5P_6 \sqsubseteq G_j^2$, the new edges ($P_4P_7$ and $P_7P_5$) and faces ($\Delta P_4P_7P_6$ and $\Delta P_6P_7P_5$) must be classified on $G_i^1$ and $G_j^2$, respectively.

(2) In Fig. 5a and c, the new vertex $P$ must inherit the classifications of $P_1$ and $P_2$. If $P_1 \sqsubseteq G_i^0$, then $P \sqsubseteq G_i^0$; and if $P_2 \sqsubseteq G_j^0$, then $P \sqsubseteq G_j^0$. In Fig. 5b, the new edge $PP_3$ must inherit the classifications of $P_1P_3$ and $P_2P_3$. If $P_1P_3 \sqsubseteq G_i^1$, then $PP_3 \sqsubseteq G_i^1$; and if $P_2P_3 \sqsubseteq G_j^1$, then $PP_3 \sqsubseteq G_j^1$. Note that an edge or facet can have more than one classification at this stage.

(3) In Fig. 6a and b, the filling operation introduces three new edges ($P_1P_3$, $P_1P_2$ and $P_1P_4$) and two new facets ($\Delta P_3P_1P_2$ and $\Delta P_1P_4P_2$). Their classifications will be determined later (see Section 5.2.3 for more details).

#### 5.2.2. Repairing the B-rep: two examples

Before examining the algorithm for repairing the B-rep, two examples are presented below to illustrate the basic idea.

In Fig. 7a, $G_1^1$ and $G_2^1$ are boundary curves of two faces $G_1^2$ and $G_2^2$. The discrete duals of the two curves are shown in Fig. 7b:

$$\begin{cases} M_1^1 = \{M_{1i}^1 | M_{1i}^1 \sqsubseteq G_1^1, i = 1, 2, \cdots, n_1\} \\ M_2^1 = \{M_{2i}^1 | M_{2i}^1 \sqsubseteq G_2^1, i = 1, 2, \cdots, n_2\} \end{cases}.$$

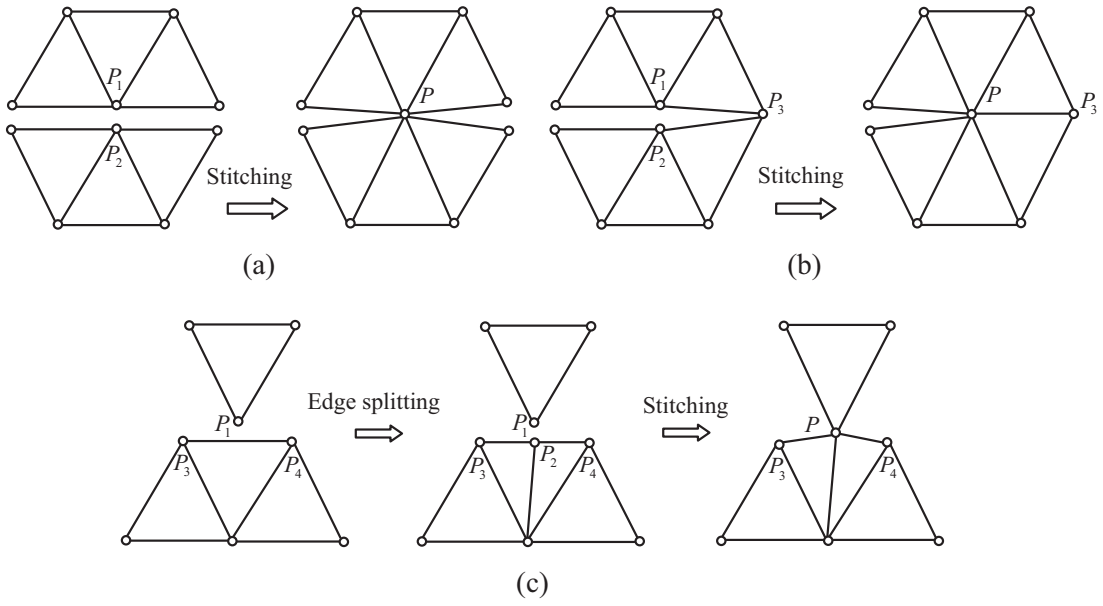


**Fig. 5.** Three cases of stitching. (a) Two vertices ($P_1$ and $P_2$) are contracted. (b) Two vertices ($P_1$ and $P_2$) and two edges ($P_1P_3$ and $P_2P_3$) are contracted. (c) Edge splitting is executed before stitching in the case of a T-connection.

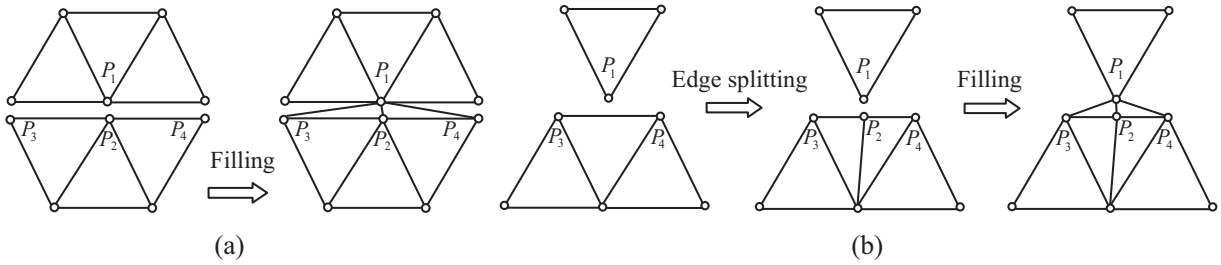


**Fig. 6.** Two cases of filling. The case (b) is a T-connection, where edge splitting is executed before filling.

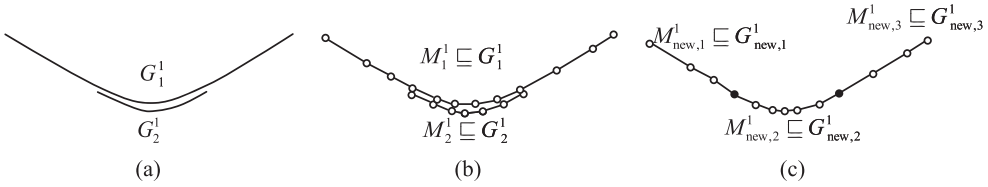


**Fig. 7.** Repairing the B-rep after stitching. (a) The initial B-rep. (b) The initial discrete model. (c) The repaired model.

It is assumed that stitching is employed to repair the discrete model because the two curves are sufficiently close from each other. After repairing, the discrete duals of the two curves are refreshed, where $M_2^1$ is a subset of $M_1^1$, as shown in Fig. 7c:

$$\begin{cases} M_1^1 = \{M_{1i}^1 | M_{1i}^1 \sqsubseteq G_1^1, i = 1, 2, \cdots, m\} \\ M_2^1 = \{M_{2i}^1 | M_{2i}^1 \sqsubseteq G_2^1, i = m_1, \cdots m_2, 1 < m_1 < m_2 < m\} \end{cases}.$$

To repair the B-rep, three curves $G_{\text{new},j}^1 (j = 1, 2, 3)$ are created. Their discrete duals are:

$$\begin{cases} M_{\text{new},1}^1 = \{M_{1i}^1 | M_{1i}^1 \sqsubseteq G_{\text{new},1}^1, i = 1, 2, \cdots, m_1\} \\ M_{\text{new},2}^1 = \{M_{1i}^1 | M_{1i}^1 \sqsubseteq G_{\text{new},2}^1, i = m_1, \cdots, m_2\} \\ M_{\text{new},3}^1 = \{M_{1i}^1 | M_{1i}^1 \sqsubseteq G_{\text{new},3}^1, i = m_2, \cdots, m\} \end{cases}.$$

After replacing $G_2^1$ with $G_{\text{new},2}^1$ and $G_1^1$ with a combination of $G_{\text{new},j}^1 (j = 1, 2, 3)$, it is observed that $G_1^2$ is adjacent to $G_2^2$ through a common curve $G_{\text{new},2}^1$ in the repaired B-rep.

Fig. 8 presents another example where the gap between the curves $G_1^1$ and $G_2^1$ are so large that filling is required to repair the discrete model. The discrete duals of the two curves are shown in Fig. 8b:

$$\begin{cases} M_1^1 = \{M_{1i}^1 | M_{1i}^1 \sqsubseteq G_1^1, i = 1, 2, \cdots, m_1\} \\ M_2^1 = \{M_{2i}^1 | M_{2i}^1 \sqsubseteq G_2^1, i = 1, 2, \cdots, m_2\} \end{cases}.$$

After repairing, a set of facets are inserted:

$$M_{\text{new}}^2 = \{M_i^2 | i = 1, 2, \cdots, n\}.$$

Instead of classifying $M_{\text{new}}^2$ onto $G_1^2$ or $G_2^2$, a face $G_{\text{new}}^2$ is created in the B-rep and $M_{\text{new}}^2$ is classified onto $G_{\text{new}}^2$:

$$M_{\text{new}}^2 = \{M_i^2 | M_i^2 \sqsubseteq G_{\text{new}}^2, i = 1, 2, \cdots, n\}.$$

Initially, a closed curve $G_{\text{new}}^1$ is created to bound $G_{\text{new}}^2$, and its discrete dual is:

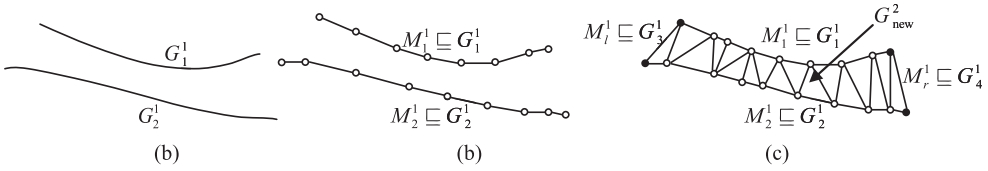


**Fig. 8.** Repairing the B-rep after filling. (a) The initial B-rep. (b) The initial discrete model. (c) The repaired model.

$M_{\text{new}}^1 = M_1^1 \cup M_2^1 \cup \{M_l^1\} \cup \{M_r^1\}$,

where $M_l^1$ and $M_r^1$ are two boundary edges of $G_{\text{new}}^2$, as shown in Fig. 8c. Furthermore, $G_{\text{new}}^1$ is subdivided into four curves $G_k^1 (k = 1 \sim 4)$, where $G_3^1$ and $G_4^1$ are new curves with $M_l^1 \sqsubseteq G_3^1$ and $M_r^1 \sqsubseteq G_4^1$. In the repaired B-rep, $G_{\text{new}}^2$ is adjacent to $G_1^2$ and $G_2^2$ through $G_1^1$ and $G_2^1$, respectively.

*5.2.3. Repairing the B-rep: the algorithm*

The algorithm is a top-down procedure, i.e., targeting faces first, followed by curves and points. As explained in Section 5.2.1, new facets created by filling are not classified initially. During this stage, new faces are created in the B-rep, and unclassified facets will be classified on these faces, as illustrated in Algorithm 1.

**Algorithm 1**. Create new faces in the B-rep

$M^2$: all of the unclassified facets, $M^2 = \{M_1^2, \cdots, M_n^2\}$
1. Initialise the visit flags of all facets to be 0
2. $k = 0$
3. **for** $i = 1$ **upto** $n$
4. **if** $M_i^2$ is visited
5. **continue**
6. $k$++
7. Flag $M_i^2$ as visited, and push $M_i^2$ into a stack $\Re$
8. Create a new face $G_k^2$ in the B-rep, and classify $M_i^2$ on $G_k^2$
9. **while** $\Re$ is not empty
10. Pop the top element of $\Re$, i.e., $M_t^2$
11. Access the neighbours of $M_t^2$, i.e., $M_{a1}^2$, $M_{a2}^2$ and $M_{a3}^2$:
12. **for** $m = 1$ **upto** 3
13. **if** $M_{am}^2$ is *unvisited* and *unclassified*
14. classify $M_{am}^2$ on $G_k^2$
15. push $M_{am}^2$ into $\Re$
16. Flag $M_{am}^2$ as visited

The curves of the B-rep need to be refreshed. First, a closed boundary curve is created for each new face. The boundary curve of a new face only has a discrete representation initially, which is composed of the boundary edges of the discrete model classified on the face. Then, a *curve splitting* procedure is introduced to *glue* adjacent faces in the B-rep, as detailed in Algorithm 2.

It is worth to give more explanations for Lines 21 and 24 in Algorithm 2. In Fig. 7, the curve $G_2^1$ is replaced by a new curve $G_{\text{new},2}^1$ because they have identical discrete representations. However, the curve $G_1^1$ is split by $G_{\text{new},2}^1$ into three curves (including $G_{\text{new},2}^1$) because the discrete dual of $G_{\text{new},2}^1$, i.e., $M_{\text{new},2}^1$ overlaps with the intermediate part of the dual of $G_1^1$, i.e., $M_1^1$. Instead, if either the end vertex of $M_{\text{new},2}^1$ is identical to an end vertex of $M_1^1$, $G_1^1$ should be split into two curves. In Line 24, the splitting results of $G_{i1k}^1$ may be new curves or existing curves in $G^1$ ($G_{\text{new}}^1$ is always a new curve). Here, a curve is *new* if and only if its discrete representation is unique in the B-rep.

After executing Algorithm 2, the topological curves of the B-rep are repaired. Finally, the topological points of the B-rep are updated to ensure all end points of curves are included in the B-rep.

In the above repairing process, the geometry entities and the relationship between geometry and topology entities must be refreshed as well. The trivial details regarding how to implement this refreshing procedure are not presented here.

To sum up, the repaired model has the following unique features:

(1) The B-rep topology is correct.
(2) The discrete model is valid in terms of both topology and geometry.
(3) Each B-rep topology entity has a unique discrete representation.
(4) Some B-rep topology entities may have no continuous representation, e.g., the curves $G_3^1$ and $G_4^1$ and the face $G_{\text{new}}^2$ shown in Fig. 8c.
(5) Some B-rep topology entities may have more than one continuous representation, e.g., the curve $G_{\text{new},2}^1$ in Fig. 7c.

**Algorithm 2**. The curve splitting procedure that glues adjacent faces in the B-rep

1. $G^1$: all curves in the B-rep $G^1 = \{G_1^1, \cdots, G_n^1\}$
2. Initialise the masks of all curves to be 0
3. **while** the set of unmasked curves is not empty
4. $G_i^1$: an unmasked curve in $G^1$
5. $M_i^1$: $M_i^1 = \{M_{ij}^1 | M_{ij}^1 \sqsubseteq G_i^1, j = 1, \cdots, m\}$
6. **if** $m \leqslant 1$
7. Mask $G_i^1$
8. **continue**
9. $M_{ij,0}^0, M_{ij,1}^0$: the start and end vertices of $M_{ij}^1$, where $M_{ij,1}^0 = M_{ij+1,0}^0$
10. $G_{ij}^1 : G_{ij}^1 = \{G_{ijk}^1 | M_{ij}^1 \sqsubseteq G_{ijk}^1, k = 1, \cdots, n_{ij}\} (j = 1, \cdots, m)$
11. **for** $j = 2$ **upto** $m$
12. **if** $G_{ij}^1 \neq G_{i1}^1$
13. **break**
14. **if** $j > m$
15. Mask $G_i^1$
16. **continue**
17. Create a new curve $G_{\text{new}}^1$: $RCS(G_{\text{new}}^1) = M_{\text{new}}^1 = \{M_{ik}^1 | M_{ik}^1 \sqsubseteq G_i^1, k = 1, \cdots, j-1\}$
18. Mask $G_{\text{new}}^1$
19. **for** $k = 1$ **upto** $n_{i1}$
20. $M_{i1k}^1 = RCS(G_{i1k}^1)$
21. **if** $M_{i1k}^1 = M_{\text{new}}^1$
22. Replace $G_{i1k}^1$ with $G_{\text{new}}^1$ in the B-rep
23. **else if** $M_{i1k}^1$ is a superset of $M_{\text{new}}^1$
24. Split $G_{i1k}^1$ into 2∼3 curves (one of them is $G_{\text{new}}^1$)
25. Replace $G_{i1k}^1$ with its splitting results
26. Insert new curves into $G^1$ with their masks initialised to be 0

## 6. Surface defeaturing and meshing

Both the discrete model and the B-rep topology are valid after repairing, which allows two types of defeaturing and meshing strategies to be applied on the repaired model. Similar to Quadros et al. [18], one strategy is to simplify the discrete model with edge-collapse operations and then mesh the simplified discrete model. The other strategy is to combine neighbouring faces in a valid B-rep topology and then mesh the simplified B-rep. In this study, we choose the second strategy to demonstrate the benefit of having the valid B-rep topology. This approach is more suitable for mesh generation that requires high geometry fidelity for the surface mesh.

### 6.1. Surface defeaturing

Some features may prevent the generation of a *high-quality* surface mesh. Here, a mesh is identified as *low quality* if it contains many elements that are badly shaped or unnecessarily small. Fig. 9 presents various features that affect surface mesh quality, i.e., short curves, tiny faces, curve proximities and small angles. The meshes around these features have to be densified to avoid the generation of badly shaped elements. Even so, the quality of some elements may still be unacceptable, such as those around the small angle shown in Fig. 9d.

A simple and efficient method to suppress these features is *face clustering* [6,7]. For instance, if $S_1$ and $S_2$ in Fig. 9c are combined, the proximity features will be removed. As reviewed in Section 2, two approaches are applicable to combine faces: the geometry-based approach and the topology-based approach. The latter is adopted to avoid tedious and expensive computation for continuous representation of faces.

Topologically, we can represent face adjacencies by an adjacency graph, as shown in Fig. 10, where graph nodes refer to faces and graph arcs connect neighbouring faces. A face-clustering operation is equivalent to an arc-contraction operation on the adjacency graph. Based on this equivalence, Algorithm 3 presents the defeaturing workflow, which is a variant of the algorithm proposed by Inoue et al. [6].

To meet requirements from different applications, the user is allowed to mask some arcs as *internal* or *boundary*. All internal arcs are contracted in a preprocessing step (Line 1 of Algorithm 3). Geometry criteria, termed as *geometry indices*, are introduced to score unmasked arcs so that the top-scored arc can be contracted

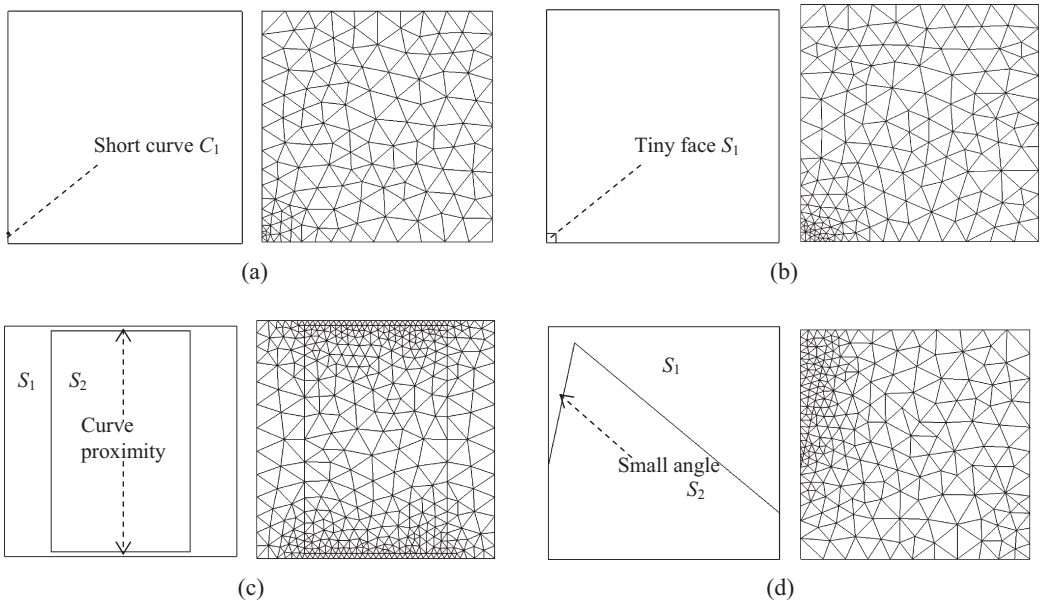


**Fig. 9.** Various features that influence surface mesh quality: (a) short curves; (b) tiny faces; (c) curve proximities; and (d) small angles.

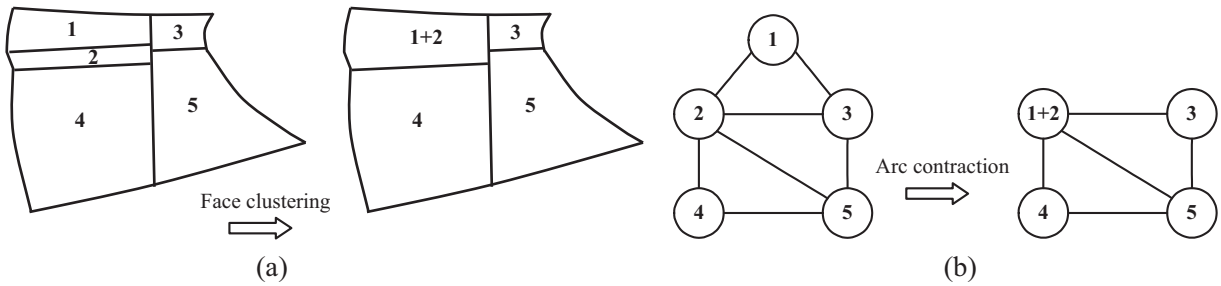


**Fig. 10.** Conversion from faces to an adjacency graph. (a) A set of faces and a face-clustering operation. (b) An adjacency graph and an arc-contraction operation.

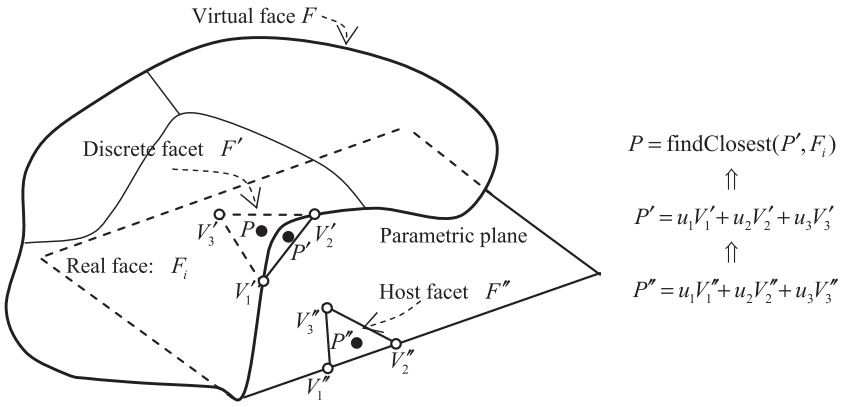


**Fig. 11.** The projection procedure for an interior mesh point in a virtual face ($F$). The point is initially located on the parametric plane, i.e., $P''$. Its projection on the discrete model is $P'$. $F''$ and $F'$ are the host facets of $P''$ and $P'$. If $F'$ is classified on a real face $F_i$ referred by $F$, the final position $P$ is the projection of $P'$ on $F_i$.

first. In addition, these indices can be employed to judge whether an arc is mergeable. In [6], many indices are proposed and four different combinations of these indices are tested for the scoring and mergeability. In our algorithm, the hybrid strategy that considers both the boundary smoothness and region flatness is adopted. More details of the implementation are referred to [6]. It needs to mention that most geometric indices are computed on the discrete model. For instance, one geometry index considers the ratios of face areas with a user-defined threshold. Obviously, the area of a face can be approximated by the sum area of all triangles classified on this face.

Note that the new faces created by filling operations, e.g., the face $G^2_{\text{new}}$ shown in Fig. 8c, can be merged with their neighbours in this defeaturing procedure.

**Algorithm 3**. The general flowchart of the defeaturing algorithm

$G$: the adjacency graph of the surface model
1. Contract all arcs in $G$ marked as *internal*
2. $A$, all of the unmasked arcs in $G$: $A = \{a_1, \cdots, a_n\}$
3. Score all arcs in $A$ by geometry criteria
4. $a_i$: a top-scored arc that is unmasked and *mergeable*
5. **while** $a_i$ exists
6. Contract $a_i$
7. Rescore all arcs incident to the ending nodes of $a_i$

## 6.2. Surface meshing

### 6.2.1. The general workflow

The meshing procedure is conducted on curves and faces individually. Algorithm 4 presents a bottom-up meshing procedure to maintain the mesh conformity on shared entities, i.e., common points of neighbouring curves and common curves of neighbouring faces. Note that this procedure is defined on the topology entities of the B-rep. When an entity is to be meshed, its boundary entities are checked first. If a boundary entity is unmeshed, a meshing function is called; otherwise, the existing mesh data is referred. This procedure ensures that all entities are meshed once. Therefore, if the input B-rep topology is correct, the mesh conformity can be ensured automatically.

**Algorithm 4**. The general workflow of the meshing algorithm

$G^2$: a set of faces contained in the surface model,
$G^2 = \{G^2_1, \cdots, G^2_n\}$
1. **for** $i = 1$ **upto** $n$
2. $G^1_i$: a set of boundary curves of $G^2_i$, $G^1_i = \{G^1_{i1}, \cdots, G^1_{im}\}$
3. **for** $k = 1$ **upto** $m$
4. **if** $G^1_{ik}$ is unmeshed
5. $G^0_{ik,0}$ and $G^0_{ik,1}$: the ending points of $G^1_{ik}$
6. **if** $G^0_{ik,0}$ is unmeshed
7. Mesh $G^0_{ik,0}$
8. **if** $G^0_{ik,1}$ is unmeshed
9. Mesh $G^0_{ik,1}$
10. $B^0_{ik}$: the boundary mesh of $G^1_{ik}$
11. Mesh $G^1_{ik}$ with given $B^0_{ik}$
12. $B^1_i$: the boundary mesh of $G^2_i$
13. Mesh $G^2_i$ with given $B^1_i$

### 6.2.2. Meshing procedures for points and curves

As explained in Section 5.2.3, a topology entity has one unique discrete representation in the repaired B-rep, but the continuous representations of point and curve entities may not exist or may not be unique. This complicates the meshing procedures for points and curves.

The mesh point of a point entity is located on its unique discrete dual, which is actually the average position of all continuous points referred by the point entity. The meshing procedure of a curve entity involves three cases. If the curve corresponds to a unique continuous representation, the meshing procedure is conducted on the continuous curve directly; otherwise, it is conducted on the unique discrete curve. Furthermore, in case that the curve entity refers to more than one continuous curve, a procedure is required to project meshing points, apart from two end points, from the discrete curve to the continuous ones. The final position of a projected mesh point is the average position of its projections.

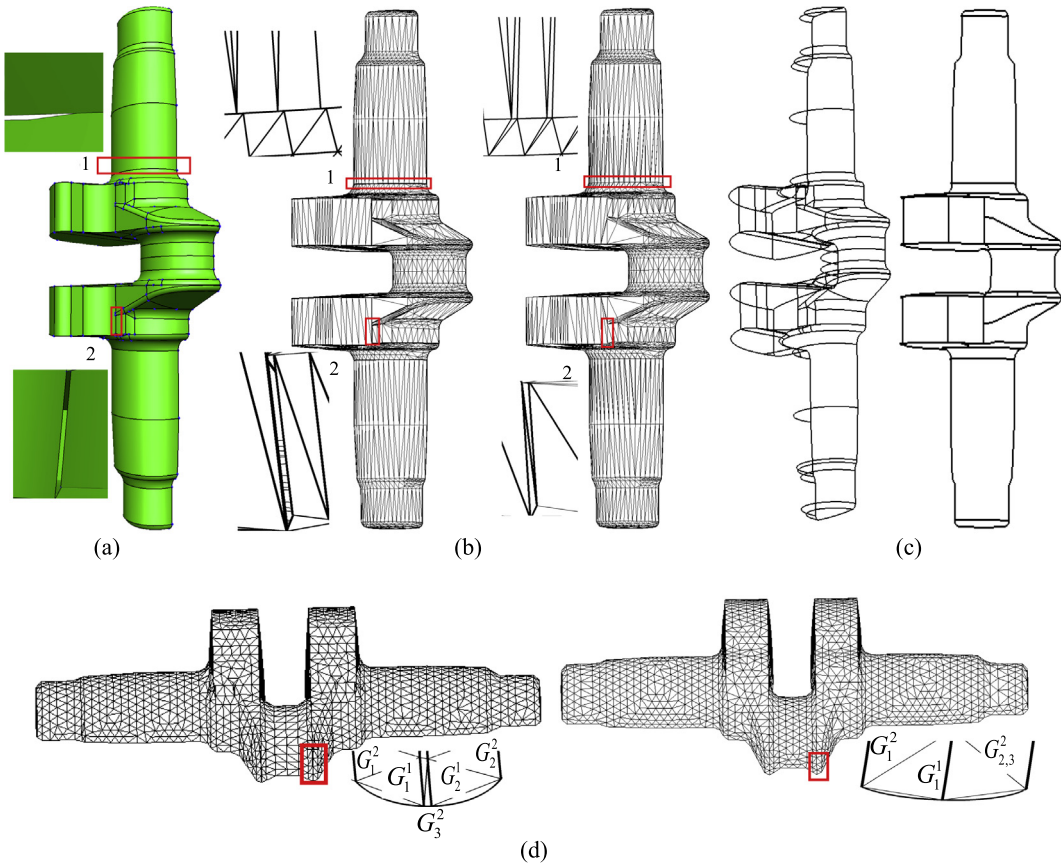


**Fig. 12.** Repairing, defeaturing and meshing results for a mechanical part model. (a) The input CAD model. (b) The initial (left) and repaired (right) discrete models. (c) The wireframes without (left) and with (right) model defeaturing. (d) The meshes without (left) and with (right) model defeaturing.

*6.2.3. Meshing procedures for faces*

There are two types of faces in the simplified B-rep after defeaturing, if all the tiny faces introduced by filling operations are merged with their neighbours. A real face corresponds to a face of the input model and has a unique continuous representation. It can be meshed in its intrinsic parametric space by a mapping-based mesher. However, a virtual face refers to a combination of faces and has no intrinsic parametric space, and the mapping-based mesher cannot be applied directly [27–29]. Therefore, an indirect approach is adopted, which meshes the discrete model of the virtual face first, and then project mesh points onto the continuous model.

The discrete model of a virtual face is represented by a group of triangular facets. By computing an average vector of all normal vectors of these facets, a parametric plane is determined perpendicular to this average normal vector. The mesh is firstly generated on the parametric plane. Its interior points are then projected onto the discrete model, and finally projected onto the continuous model if possible. Fig. 11 illustrates the projection procedure. A parametric mesh point $P''$ is contained in a host facet $F''$. $P'$ is the projection of $P''$ on the discrete model, contained by a facet $F'$. Here, $F''$ is the projection of $F'$ on the parametric plane. Assuming that the area coordinates of $P'$ in $F'$ and $P''$ in $F''$ are equal, we can interpolate the position of $P'$ using three vertices of $F'$. If $F'$ is classified on a real face $F_i$ referred by $F$, the final position $P$ is the projection of $P'$ on $F_i$.

To rectify the mesh validity and quality issue due to projection distortion, the following strategies are adopted:

(1) A small geometry tolerance is given when tessellating the input continuous model so that the discrete model is close enough to the input model.
(2) Flatness of a virtual face is considered when defeaturing so that the parameterisation approach for the discrete model does not induce too large projection distortion between the parametric plane and the discrete model.
(3) A Riemann metric tool is employed when generating the mesh on the parametric plane.

## 7. Results

The proposed surface repairing, defeaturing and meshing algorithms have been implemented and tested on many examples. Fig. 12 presents some results by employing these algorithms on a mechanical part model. Fig. 12a and b show the continuous input model and its discrete counterpart, respectively. As shown in Fig. 12a, small gaps and overlaps exist in the input model. After tessellating the model to a discrete one, these errors are inherited, as shown in the left of Fig. 12b. The extended B-rep records both the continuous and discrete models. Moreover, the mappings of the B-rep and the discrete model are initialised and recorded. To tackle the geometry errors, the discrete model is repaired first. The result is shown in the right of Fig. 12b, where the errors of the discrete model are resolved. Different with the algorithm proposed by Patel et al. [15,16], where the repaired discrete model is sent to a surface mesher directly, our algorithm continues to repair the B-rep. The left of Fig. 12c illustrates the repaired B-rep, where the boundary curves of all faces are rendered. The repaired model can be directly sent to the meshing procedure presented in Algorithm 4 to get a conformal surface mesh, as shown in the left of Fig. 12d. However, the repaired model contains many features, including those defined on the input model and those introduced by the repairing algorithm. The left of Fig. 12d reveals the mesh

**Table 1**
Quality comparison of the meshes without and with model defeaturing.

| Indices | | Mechanical part | | Flying minnow | |
|---|---|---|---|---|---|
| | | Without | With | Without | With |
| No. of elems. | | 3327 | 3192 | 17,660 | 16,608 |
| $\alpha_{min}$ (°) | | 0.01 | 13.4 | 0.93 | 12.1 |
| $\alpha_{max}$ (°) | | 176.5 | 143.2 | 169.4 | 138.5 |
| $\alpha_{ave}$ (°) | | 47.5 | 52.2 | 47.9 | 49.5 |
| Distributions of $\alpha_{min}$ (No. of elems and percentage) | $0 < \alpha_{min} \leqslant 5°$ | 61(1.83) | 0 | 14(0.08) | 0 |
| | $5° < \alpha_{min} \leqslant 10°$ | 67(2.01) | 0 | 17(0.10) | 0 |
| | $10° < \alpha_{min} \leqslant 15°$ | 26(0.78) | 4(0.13) | 73(0.41) | 9(0.05) |
| | $15° < \alpha_{min} \leqslant 20°$ | 20(0.60) | 5(0.16) | 91(0.52) | 37(0.22) |
| | $20° < \alpha_{min} \leqslant 25°$ | 28(0.84) | 0 | 141(0.80) | 58(0.35) |
| | $25° < \alpha_{min} \leqslant 30°$ | 48(1.44) | 26(0.81) | 343(1.94) | 217(1.31) |
| | $0° < \alpha_{min} \leqslant 30°$ | 250(7.5) | 35(1.10) | 679(3.85) | 321(1.93) |

details on a narrow face introduced by filling operations, where highly stretched elements are generated because the local proximity distance is far smaller than mesh sizes.

To address this issue, the B-rep is defeatured before meshing. The right of Fig. 12c renders the wireframe of the defeatured model. After defeaturing, the number of faces decreases from 104 to 26. More importantly, the quality of the mesh generated on the defeatured model is much better. For instance, the small face shown in the left of Fig. 12d is merged with its neighbours after defeaturing, which improves local element shapes, as shown in the right of Fig. 12d.

Note that the size maps for the two meshes shown in Fig. 12d are similar. Therefore, the element numbers of these two meshes are very close, containing 3327 and 3192 elements, respectively. Table 1 compares the quality of the two meshes. If the minimal interior angle ($\alpha$) of an element is less than 30°, the element is identified as *low-quality*. The mesh without model defeaturing contains 250 low-quality elements, about 7.5% of the total elements. The minimal and maximal interior angles ($\alpha_{min}$ and $\alpha_{max}$) of all elements are 0.01° and 176.5°, respectively; the average angle of all $\alpha$ values ($\alpha_{ave}$) is 47.5°. With model defeaturing, all indices are improved remarkably. The mesh contains only 35 low-quality elements, about 1.1% of the total number; $\alpha_{min}$, $\alpha_{max}$ and $\alpha_{ave}$ are improved to 13.4°, 143.2° and 52.2°, respectively.

Fig. 13 presents the results for a flying minnow model, which was also used by Patel et al. [15,16] to demonstrate their repairing algorithm for discrete models. The input model is supposed to be closed. However, many boundary edges are detected on the initial discrete model, which are painted in thick lines in the left of Fig. 13b. After repairing, no boundary edges are detected, as shown in the right of Fig. 13b. This means that the repaired discrete model is conformal and watertight. Besides, the initial and defeatured models and the meshes on both models are compared in Fig. 13c and d, respectively. Table 1 lists the quality data of the two meshes. With model defeaturing, remarkable improvement are observed as well. The mesh of the initial model contains 679 low-quality elements, about 3.85% of the total number. $\alpha_{min}$, $\alpha_{max}$ and $\alpha_{ave}$ are 0.93°, 47.9° and 169.4°, respectively. With model defeaturing, the mesh contains only 321 low-quality elements, about 1.93% of the total number; $\alpha_{min}$, $\alpha_{max}$ and $\alpha_{ave}$ are improved to 12.1°, 49.5° and 138.5°, respectively.

In addition, Fig. 13e enlarges some local mesh details. In the initial model, $G_1^1$ and $G_2^1$ are boundary curves of a tiny face $G_2^2$. Besides the proximity feature between these two curves, small angles are defined on the corners where the curves meet. After defeaturing, these features are removed, and as a result the element quality is improved significantly.

Finally, given a CRH (China Railway High-speed) train model (Fig. 14a), three different meshing approaches are compared.

(1) The proposed approach first repairs simultaneously the tessellated model and the B-rep and then simplifies the repaired B-rep through *virtual* face combination. The final mesh is generated on the simplified B-rep and the mesh points are projected back to the continuous input geometry.
(2) The second approach is adopted by many commercial tools [10]. It first repairs directly the gaps of the input model by redefining the curves and surfaces in the neighbourhood and then defeatures the repaired model through *real* face combination. The continuous geometry is modified in both steps and the final mesh is generated on this *modified* geometry.
(3) The third approach repairs the tessellated model by using the algorithm proposed by Patel et al. [15,16], and then meshes the repaired discrete model [27]. Because the B-rep of the input model is lost after tessellation, the meshing procedure needs to rebuild the geometry features of the discrete model. With these feature edges and some other mesh edges, the discrete model is subdivided into many *meshable patches*.

To facilitate the comparison, the defeatured model in the second approach has the same B-rep structure as that in the first approach. Meanwhile, an equal element size value (i.e., 0.15) is used in the three meshing approaches to ensure the meshes have nearly similar element numbers.

Fig. 14b enlarges a geometric error of the input CAD model and its repairing result using the second approach. This error is inherited after tessellation, and both the first and third approaches need to repair this error in the discrete model. Fig. 14c enlarges the inherited error and its repairing result.

Both the first and second approaches obtain a valid B-rep after repairing, and need to simplify this B-rep by combining neighbouring faces. Differently, the first approach modifies only the topology using virtual operations while the second approach modifies both topology and geometry using real operations. Fig. 15a and b render the simplified B-rep on the continuous and discrete models, respectively. However, the third approach does not output a valid B-rep after repairing and cannot employ the B-rep based defeaturing algorithm.

Fig. 16 presents the surface meshes generated through different approaches. Both meshes output by the first and the second approaches respect the simplified B-rep. Meanwhile, the first approach does not modify the continuous geometry in the repairing and defeaturing steps; therefore, the mesh points are located on the input CAD model exactly. In contrast, the mesh points output by the second approach are located on a modified CAD model. Different with the first and second approaches, the third approach generates the mesh on the discrete model directly. The mesh it

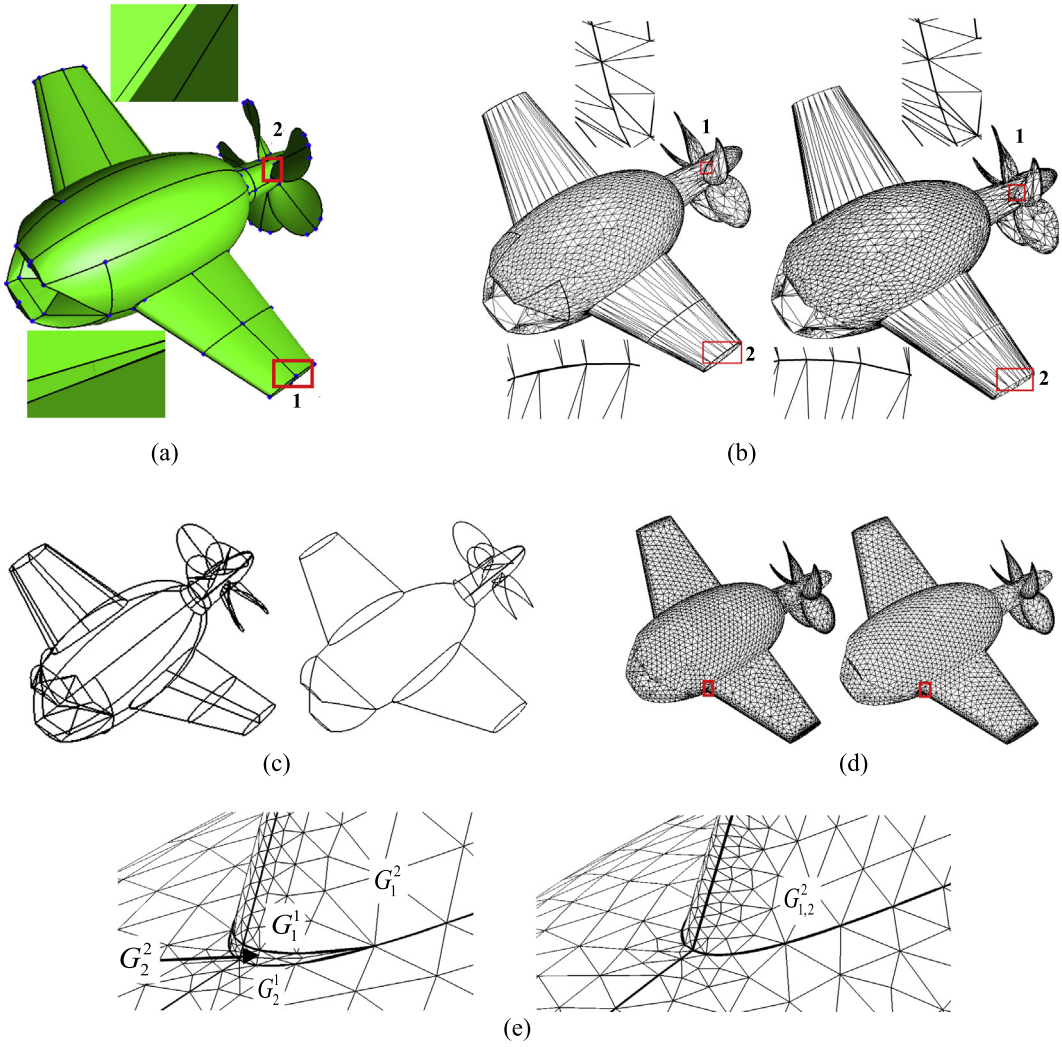


**Fig. 13.** Repairing, defeaturing and meshing results for a flying minnow model. (a) The input CAD model. (b) The initial (left) and repaired (right) discrete models. (c) The wireframes without (left) and with (right) model defeaturing. (d) The meshes on the initial (left) and defeatured (right) models. (e) Local details of the meshes shown in (d).

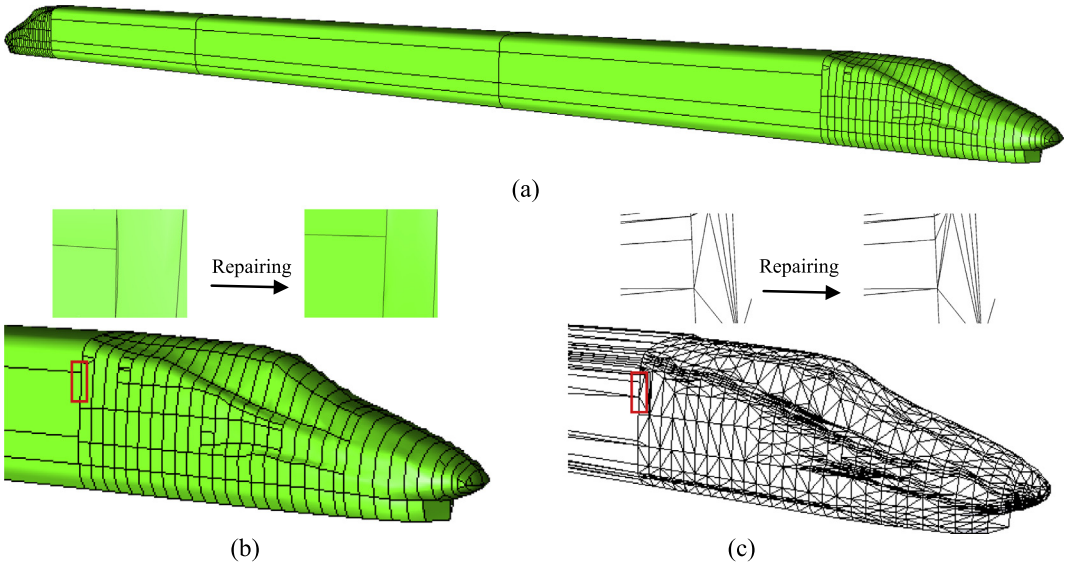


**Fig. 14.** Repairing results of the CRH train model. (a) is the input CAD model; (b) shows a geometric error in the input model and its repairing result. This error is inherited after tessellation. (c) shows this error in the discrete model and its repairing result.

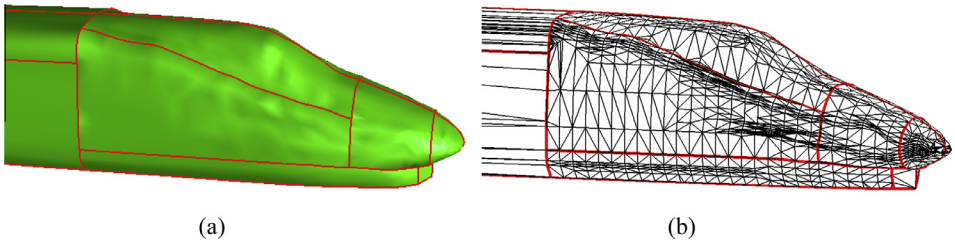


**Fig. 15.** The simplified B-rep of the train model on (a) the continuous and (b) discrete models.

outputs does not respect the simplified B-rep, and many mesh lines cross over the boundary curves, as shown in Fig. 16c.

Although the element shapes of the three meshes shown in Fig. 16 are all qualified for downstream volume meshing and numerical simulations, the quality data of the three meshes in terms of the geometry accuracy are quite different, which may have a significant impact on the simulation accuracy. Here, two scalar variables are calculated to evaluate the geometry accuracy of a surface mesh. They are respectively named as the *node distance* and *element distance*, evaluating how far a node or an element of the mesh deviates from the continuous input model. The node distance is the minimal value of the distances between a node and different surface patches of the input model, and the element distance is approximated by the maximal values of the node distances of the forming points and centroid point of an element.

The node distances are mainly determined by how far the geometry on which the surface mesh is generated deviates from the input geometry. The proposed approach ensures the mesh points are loyal to the input geometry; therefore, all node distances are zero. However, the second approach modifies the input geometry in both repairing and defeaturing steps. As a result, nonzero node distances exist in the region where the geometry is modified, and the maximal node distance is 0.075. The third approach generates the mesh on the repaired discrete model. The node distance sources from both the tessellation and repairing steps, and its maximal value is 0.16. Compared to the average length of mesh edges (about 0.15), the node distances of the meshes output by the second and third approaches are clearly noticeable, as presented in Fig. 17a and b.

The mesh elements of a curved surface deviate from the surface significantly even when the mesh nodes are exactly located on the surface. For the proposed approach, the element distance mainly depends on the element size and surface curvature. However, for the other two approaches, the element distance is also impacted by the node distance. Fig. 18b–d present the colour maps of the element distance for the three meshes shown in Fig. 16. To highlight the difference, the head part of the train is enlarged for comparison, and the values represented by the colour legends are clipped into [0,0.06].

Because a uniform mesh size is specified, the variation of the element distance for the mesh output by the proposed approach is mainly related to the surface curvature. Fig. 18a renders a colour map for the surface curvature, where a curvature value is defined for each element, and this value is approximated by the average value of the curvatures of the three forming points of the element. To improve the geometry fidelity of this mesh further, adaptive meshing techniques [30] can be employed to set small element sizes at high curvature regions, or curved elements instead of planar elements can be used to compose the mesh [31].

Fig. 19 compares the distributions of element distance for the three meshes. Because planar elements always deviate from curved surfaces due to the surface curvature, the elements deviating from the input geometry by less than 0.01 are excluded from the comparison to highlight the difference of the three approaches. It is evident that the mesh output by the third approach deviates from the input geometry most, where the maximum element distance is 0.16, and about 64.1% elements deviate from the input geometry by more than 0.01. This also explains why many simulations that require meshes with high geometry fidelity do not adopt the meshes defined on discrete models as inputs. The second approach defines the mesh on the modified CAD model and improves the geometry accuracy of the output mesh remarkably,

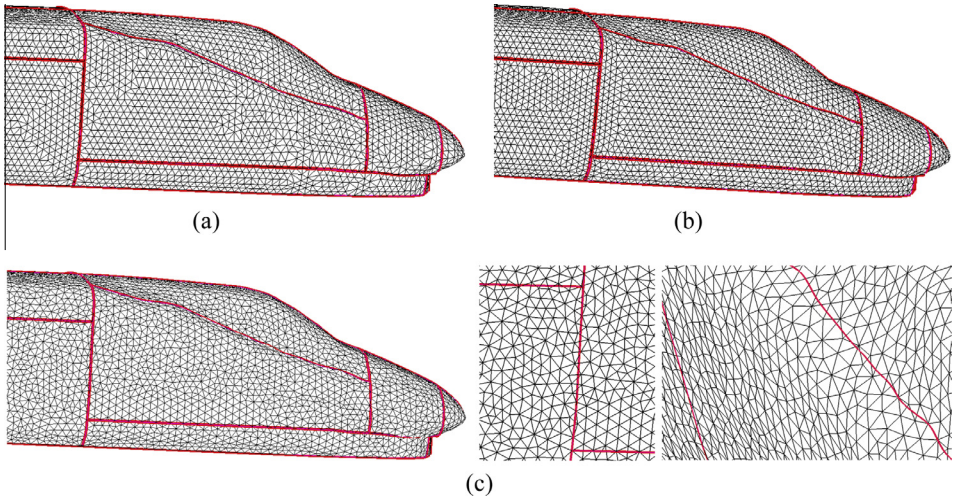


**Fig. 16.** The surface meshes of the train model. (a)–(c) show the meshes output by the first, second and third approaches, respectively. In addition, (c) presents two closeup views of the third mesh.

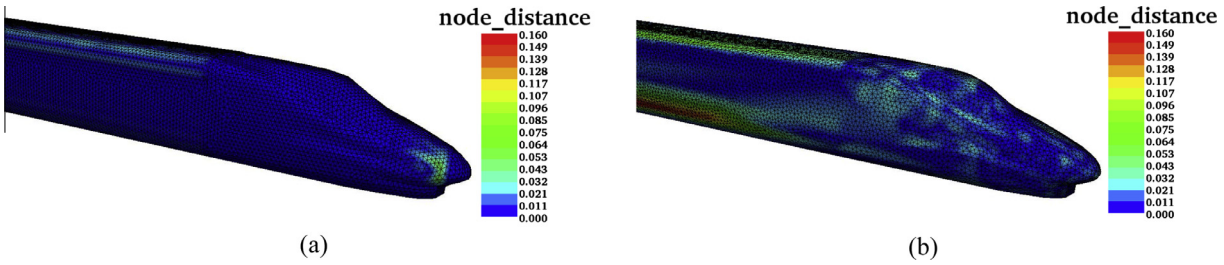


**Fig. 17.** The colour maps of the node distances of the meshes output by (a) the second and (b) third approaches. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

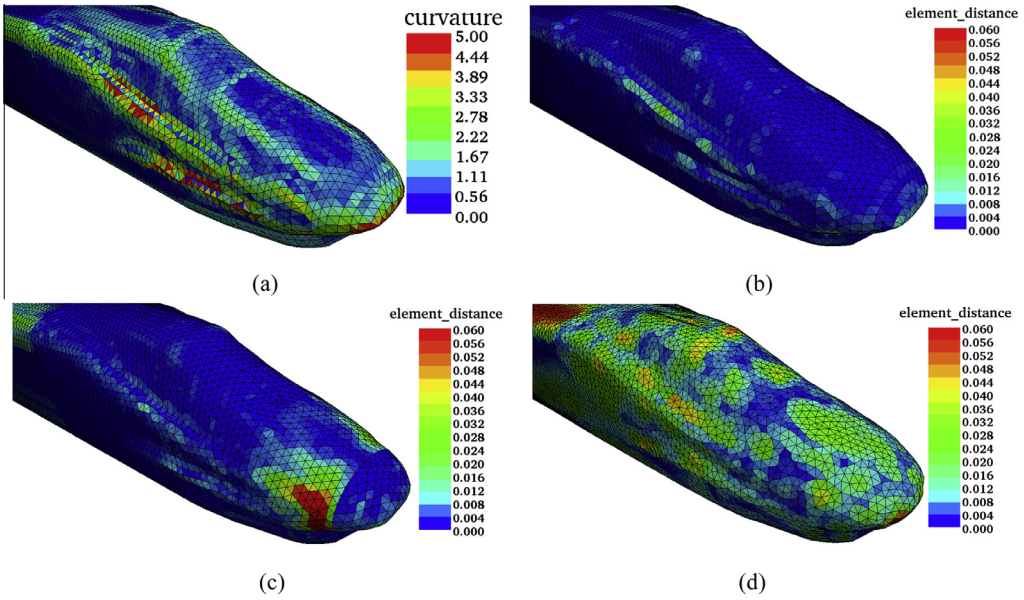


**Fig. 18.** (a) is the colour map for the surface curvature defined on the mesh output by the first approach. (b)–(d) are the colour maps of the element distance for the meshes output by the first, second and third approaches, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

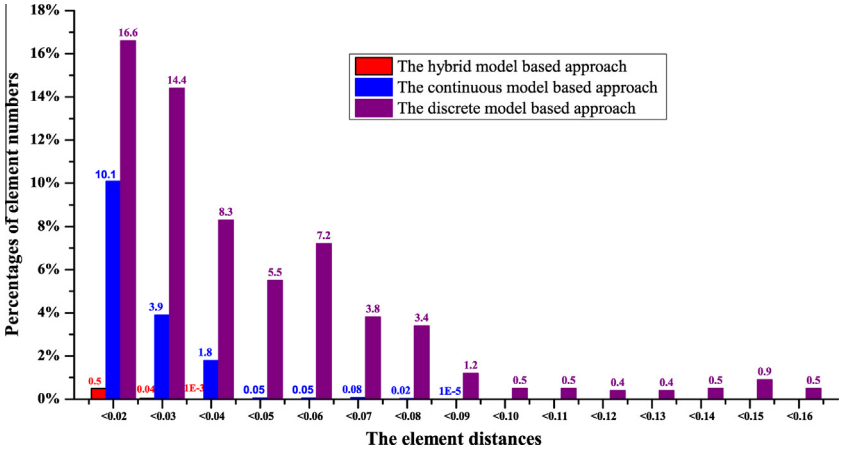


**Fig. 19.** The distributions of element distance for the meshes output by different approaches.

where the maximum element distance is 0.081, and about 16.0% elements deviate from the input geometry by more than 0.01. Not surprisingly, the proposed approach outputs the best mesh in terms of the geometry accuracy, where the maximum element distance is reduced to 0.0302, and only about 0.5% elements deviate from the input geometry by more than 0.01.

## 8. Conclusions

A new data structure named hybrid surface B-rep is proposed to represent a composite surface model. In this extended B-rep, each topology entity has a dual geometric representation in default, i.e., a continuous representation and its discrete counterpart. The

inclusion of dual geometry data enables a sequence of repairing, defeaturing and meshing algorithms for an input CAD model with small gaps and overlaps on its surface boundaries. The continuous input geometry remains untouched in the entire process, and the output mesh is loyal to the input model. This feature is desirable in many simulations that require meshes with high geometry fidelity. Meanwhile, most geometry computations are defined on the discrete model in the proposed approach, avoiding their tedious, expensive and unreliable counterparts defined on the continuous model.

## Acknowledgements

The authors would like to thank the support from the National Natural Science Foundation of China (Grant Nos. 11172267, 11432013, 10872182 and 11272285), Zhejiang Provincial Natural Science Foundation of China (Grant No. Y1110038) and High Performance Computing Wales. The first author acknowledges the joint support from Zhejiang University and China Scholarship Council and the host of Professor Oubay Hassan and Professor Kenneth Morgan for his visiting research at Swansea University, UK.

## References

[1] Shimada K. Current issues and trends in meshing and geometric processing for computational engineering analyses. J Comput Inform Sci Eng 2011;11:021008–21013.
[2] Thakur A, Banerjee AG, Gupta SK. A survey of CAD model simplification techniques for physics-based simulation applications. CAD 2009;41:65–80.
[3] Cottrell JA, Hughes TJR, Bazilevs Y. Isogeometric analysis toward integration of CAD and FEA. John Wiley & Sons Ltd; 2009.
[4] Chen J, Zhao D, Huang Z, Zheng Y, Gao S. Three-dimensional constrained boundary recovery with an enhanced Steiner point suppression procedure. Comput Struct 2011;89:455–66.
[5] Sheffer A, Bercovier M, Blacker T, Clements J. Virtual topology operators for meshing. Int J Comput Geom Appl 2000;10:309–31.
[6] Inoue K, Itoh T, Yamada A, Furuhata T, Shimada K. Face clustering of a large-scale CAD model for surface mesh generation. CAD 2001;33:251–61.
[7] Sheffer A. Model simplification for meshing using face clustering. CAD 2001;33:925–34.
[8] Sun R, Gao S, Zhao W. An approach to B-rep model simplification based on region suppression. Comput Graph 2010;34:556–64.
[9] Kim BC, Mun D. Feature-based simplification of boundary representation models using sequential iterative volume decomposition. Comput Graph 2014;38:97–107.
[10] Gammon M. CAD clean-up for meshing. Short courses of the 23rd International Meshing Roundtable. London, UK; 2014.
[11] Ribó R, Bugeda G, Oñate E. Some algorithms to correct a geometry in order to create a finite element mesh. Comput Struct 2002;80:1399–408.
[12] Clark B. Removing small features with real CAD operations. In: Brewer M, Marcum D, editors. Proceedings of the 16th international meshing roundtable. Berlin Heidelberg: Springer; 2008. p. 183–98.
[13] John D, Robert H. Quilts: a technique for improving boundary representations for CFD. In: Proceedings of the16th AIAA computational fluid dynamics conference, Orlando, FL, USA; 2003. AIAA 2003-4131.
[14] Foucault G, Cuillière J-C, François V, Léon J-C, Maranzana R. Adaptation of CAD model topology for finite element analysis. CAD 2008;40:176–96.
[15] Patel PS, Marcum DL, Remotigue MG. Automatic CAD model topology generation. Int J Numer Method Fluid 2006;52:823–41.
[16] Patel PS, Marcum DL. Robust and efficient CAD topology generation using adaptive tolerances. Int J Numer Mech Eng 2008;75:355–78.
[17] Smith B, Tautges T, Wilson PH. Sealing faceted surfaces to achieve watertight CAD models. In: Shontz S, editor. Proceedings of the 19th international meshing roundtable. Berlin Heidelberg: Springer; 2010. p. 177–94.
[18] Quadros W, Owen S. Defeaturing CAD models using a geometry-based size field and facet-based reduction operators. In: Clark B, editor. Proceedings of the 18th international meshing roundtable. Berlin Heidelberg: Springer; 2009. p. 301–18.
[19] Haimes R, Follen G. Computational analysis programming interface. In: Proceedings of the 6th international conference on numerical grid generation in computational field simulations, London, UK; 1998. p. 663–72.
[20] CADNexus CAPRI CAE Gateway. <http://www.cadnexus.com/index.php/capri.html>. [01.03.14].
[21] Tautges TJ. The common geometry module (CGM): a generic, extensible geometry interface. In: Proceedings of the 9th international meshing roundtable, New Orleans, LA, USA; 2000. p. 337–48.
[22] The common geometry module. <https://cubit.sandia.gov/public/cgm.html>. [01.03.14].
[23] The CUBIT geometry and mesh generation toolkit. <https://cubit.sandia.gov/>. [01.03.14].
[24] Pointwise: mesh generation software for CFD. http://www.pointwise.com/.
[25] Hamri O, Léon J-C, Giannini F, Falcidieno B. Software environment for CAD/CAE integration. Adv Eng Softw 2010;41:1211–22.
[26] Dey S, Shephard MS, Flaherty JE. Geometry representation issues associated with p-version finite element computations. Comput Meth Appl Mech Eng 1997;150:39–55.
[27] Xie L, Chen J, Liang Y, Zheng Y. Geometry-based adaptive mesh generation for continuous and discrete parametric surfaces. J Inform Comput Sci 2012;9:2327–44.
[28] Foucault G, Cuillière JC, François V, Léon JC, Maranzana R. An extension of the advancing front method to composite geometry. In: Brewer M, Marcum D, editors. Proceedings of the 16th International Meshing Roundtable. Berlin Heidelberg: Springer; 2008. p. 287–314.
[29] Foucault G, Cuillière JC, François V, Léon JC, Maranzana R. Generalizing the advancing front method to composite surfaces in the context of meshing constraints topology. CAD 2013;45:1408–25.
[30] Xiao Z, Chen J, Zheng Y, Zeng L, Zheng J. Automatic unstructured element-sizing specification for surface mesh generation. Procedia Engineering, special issue on the 23nd International Meshing Roundtable, to appear. doi: 10.1016j.proeng.2014.10.387.
[31] Xie Z, Sevilla R, Hassan O, Morgan K. The generation of arbitrary order curved meshes for 3D finite element analysis. Comput Mech 2013;51:361–74.