Computers & Graphics 35 (2011) 461–470

Contents lists available at ScienceDirect

Computers & Graphics

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

SMI 2011: Full Paper

An approach to automated decomposition of volumetric mesh

Chuhua Xian, Shuming Gao *, Tianming Zhang

State Key Laboratory of CAD & CG, Zhejiang University, 310058 Hangzhou, PR China

ARTICLE INFO

Article history:
Received 10 December 2010
Received in revised form
1 April 2011
Accepted 2 April 2011
Available online 9 April 2011

Keywords:
Volumetric mesh
Mesh decomposition
Tetrahedral mesh
Hexahedral mesh
Electric flux

ABSTRACT

Mesh decomposition is critical for analyzing, understanding, editing and reusing of mesh models. Although there are many methods for mesh decomposition, most utilize only triangular meshes. In this paper, we present an automated method for decomposing a volumetric mesh into semantic components. Our method consists of three parts. First, the outer surface mesh of the volumetric mesh is decomposed into semantic features by applying existing surface mesh segmentation and feature recognition techniques. Then, for each recognized feature, its outer boundary lines are identified, and the corresponding splitter element groups are setup accordingly. The inner volumetric elements of the feature are then obtained based on the established splitter element groups. Finally, each splitter element group is decomposed into two parts using the graph cut algorithm; each group completely belongs to one feature adjacent to the splitter element group. In our graph cut algorithm, the weights of the edges in the dual graph are calculated based on the electric field, which is generated using the vertices of the boundary lines of the features. Experiments on both tetrahedral and hexahedral meshes demonstrate the effectiveness of our method.

1. Introduction

Mesh decomposition (or mesh segmentation) is the process of dividing a mesh into meaningful components. Over the last decade, mesh decomposition has become an active research topic, and it has provided a fundamental process for obtaining useful information from an input mesh in many applications, such as reverse engineering [1], shape analysis and understanding [2,3], modeling [4], compression [5], collision detection [6], and skeleton-driven animation [7]. Today, there are many methods [8,9] to decompose a mesh model. However, most of these methods aim to process triangular meshes only.

With the development of meshing techniques, volumetric mesh models have become widely used in many applications, including physically based simulation, product quality evaluation, and scientific visualization. As the magnitude of the volumetric elements increases, it is necessary to decompose the mesh model into several parts for local manipulation.

If the performance of the product does not meet a realistic requirement during the process of numerical behavior simulation, the product may need modification or addition of new parts. As discussed in [10], reusing the pre-generated meshing semantic data in existing products can reduce the time substantially. In these applications, the volumetric mesh model needs to be decomposed into semantic components to allow for convenient copying and reuse. Fig. 2 shows an example of this process. The models in (a) and (b) have been meshed into tetrahedral meshes. If we want to reuse the interesting semantic feature in (c), we first separate that feature from the original model (see (d)) and then copy and paste the feature onto the new model (see (e) and (f)). These operations make it unnecessary to re-mesh the whole model and thus save substantial time during the simulating process.

To directly edit a volumetric mesh, some elements may need to be optimized to meet the requirement of numerical simulation during the finite element (FE) analysis. The optimization process is time-consuming if the model contains a large number of elements. Generally, the affected regions are limited to some local parts. Thus, we only need to optimize these parts locally. As shown in Fig. 3, we select and separate the part from the original model to change the middle component of a volumetric mesh mechanical part. We then modify the part and optimize the elements locally. In this context, it is significant to study approaches to decomposing a volumetric mesh.

The objective of this paper is to develop a method that can decompose the volumetric mesh into semantic components required by the editing and partial reuse of volumetric meshes. The decomposition method intends to satisfy the following conditions: decompose the volumetric mesh in an automatic way, guarantee the quality of the decomposed semantic components, deal with a volumetric mesh with complex semantic

* Corresponding author.
E-mail addresses: xianchuhua@cad.zju.edu.cn, chuhuaxian@gmail.com (C. Xian), smgao@cad.zju.edu.cn (S. Gao), zhangtianming@cad.zju.edu.cn (T. Zhang).

0097-8493/$ - see front matter Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2011.04.001

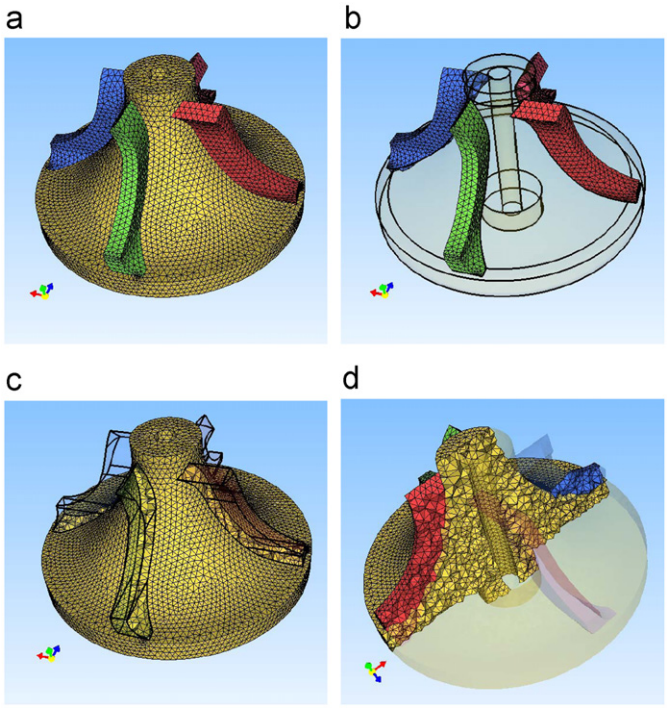

Fig. 1. Decomposition of the tetrahedral mesh of the turbo model.

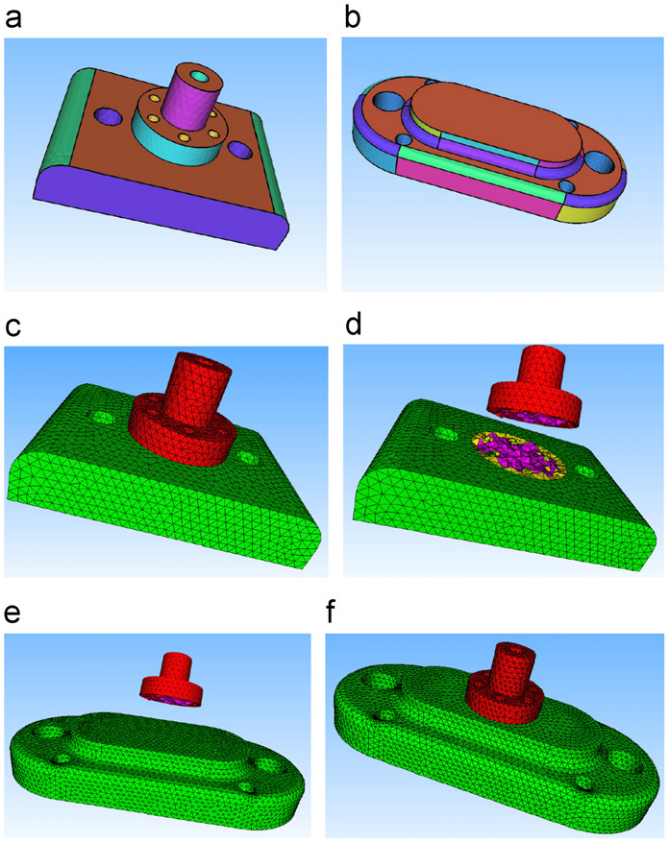

Fig. 2. Tetrahedral mesh reused. (a) and (b) are two original CAD models. The proposed method can separate the selected sematic component from the existing volumetric mesh model (c), (d) and paste that component onto another volumetric mesh model (e), (f).

components, and provide reasonable computational efficiency. To satisfy the above tasks, we first decompose the outer surface mesh of a volumetric model into semantic features using surface mesh segmentation and feature recognition techniques and then decompose the volumetric mesh into the corresponding semantic components by using the graph cut algorithm.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 presents some notation and an overview of the approach. The details of the approach are described in Section 4. Experimental results are given in Section 5, and conclusions to this paper are in Section 6.

2. Related work

Due to their wide range of applications, many 3D polygon mesh decomposition algorithms have been studied over the last decade. A review of mesh segmentation can be found in [8,9]. A more recent survey of decomposition approaches appears in [11]. In [11], Chen et al. compare some recent segmentation algorithms to ground truth based on several evaluation metrics.

Briefly, the decomposition algorithms can be categorized into two classes based on different types of input models. The first class segments natural graphical models into meaningful regions that correspond to human vision. The second class aims at processing engineering models.

Mangan and Whitaker [12] first adapted the morphological watershed segmentation algorithm from the analysis of images to the analysis of surface mesh. This method first finds the vertices with local minimal curvature, then clusters nearby vertices using the steepest decent approach. Katz et al. [13] use feature points to segment the mesh hierarchically. Their method is invariant to the pose of the model and can generate coarse levels and fine levels of segmentation. Shlafman et al. [14] apply the k-means algorithm to clustering the faces of the mesh and then to segment the mesh into meaningful pieces. In [7] Katz and Tal present an approach to improve this method to achieve hierarchical segmentation by using fuzzy clustering. Their method can segment the model in a top-down hierarchy. By adapting the mean shift, Yamauchi et al. [15] propose a scheme for generating feature sensitive segmentation. Another hierarchical decomposition algorithm is presented in [16] using a curved skeleton. Lai et al. [17] extend the random walk method from an image to a triangular mesh and obtain an efficient segmentation. The random walk-based algorithm provides results in comparable quality to other methods and is more efficient. Skraba et al. [18] combine persistence-based clustering with the Heat Kernel Signature function to obtain an isometric invariant mesh segmentation. In [19], an intuitive user interface (UI), called cross-boundary brushes, is introduced for interactive mesh decomposition. This algorithm segments the mesh based on cutting along an isoline of a harmonic field.

To segment the engineering models, Lavoué et al. [20] present an efficient algorithm that uses curvature tensor field analysis for the decomposition of arbitrary triangular meshes. Vierira et al. extend region-growing techniques to segment the unstructured noisy mesh and to extrude smooth surface. Hwan Kim et al. [21] propose an algorithm that merges the over-segmented regions, to refine the boundaries. Várady et al. apply the Morse–Smale complex to decompose the mesh into regions [22]. In [18], Sunil and Pande segment the sheet metal CAD model by classifying the triangles into coarse and dense categories and then process them by different methods. With certain modifications, the method in [17] can also be used to process the engineering models. More comparison of methods of CAD model segmentation can be found in [23].

Although there have been many algorithms for mesh decomposition, most of them concern surface polygon mesh. In [24], Attene et al. describe an algorithm to compute a hierarchy of convex polyhedra that tightly enclose a tetrahedral mesh. Their

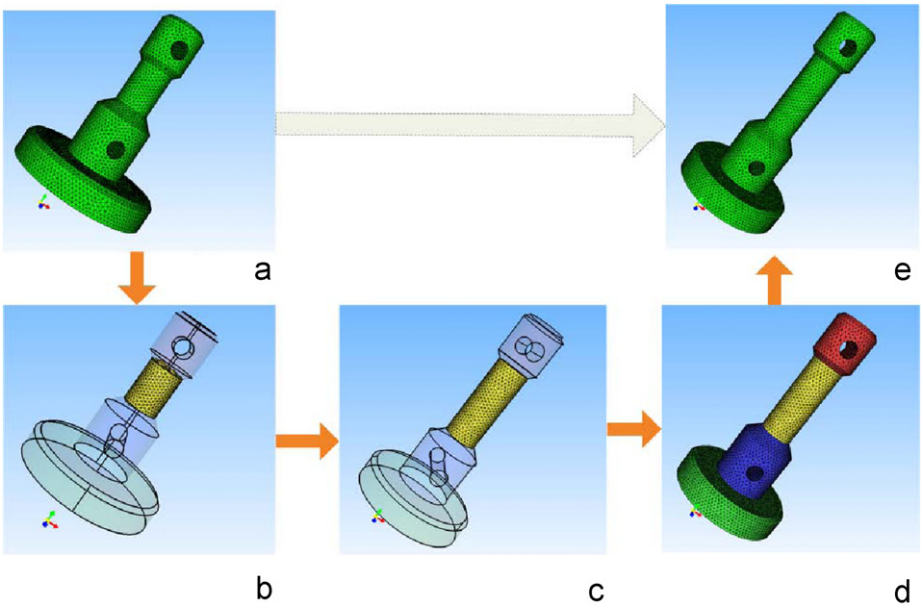

Fig. 3. Editing a tetrahedral mesh directly. (a) The tetrahedral mesh model. (b) Select the interesting part and decompose that part from the original mesh model. (c) Change the selected part and optimize the tetrahedrons locally. (d) Reconnect the edited part with the other parts. (e) The final model.

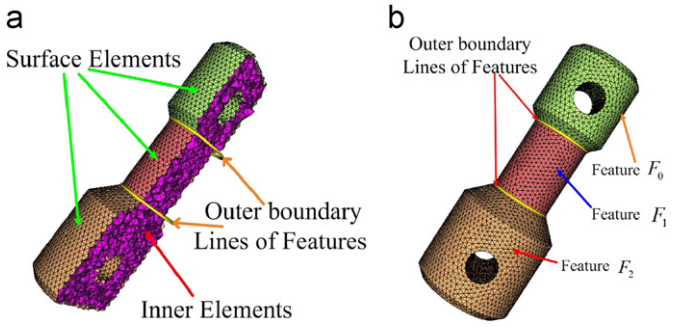

Fig. 4. Basic concepts. (a) Examples of surface elements, inner elements and *OBLF* s. (b) Examples of form features and *OBLF* s of the features.

algorithm clusters tetrahedrons hierarchically into near-convex groups and works directly on the tetrahedrons of the mesh. A limitation of their method is that the contact surface of adjacent components is not considered. In this paper, we propose a decomposition algorithm to process the volumetric mesh (tetrahedral mesh or hexahedral mesh), and our algorithm mainly aims to solve decomposition in engineering models.

3. Concepts and overview

Let M be a volumetric mesh; we then define the following concepts:

- An *element* of a volumetric mesh is a volumetric cell, such as a tetrahedron or a hexahedron in a volumetric mesh. The element is called a *surface element* if one of its faces has no adjacent element. Otherwise, the element is called an *inner element* (see Fig. 4).
- The *element group*, or *group* for short, is defined as the volumetric elements set.
- The *external surface mesh* of M consists of the faces without adjacent elements, denoted as ∂M.
- *Form feature*: we define a form feature as a semantic partial shape that has engineering meaning [25]. The form feature defined on the surface mesh ∂M consists of surface polygon faces, while the form feature defined on the volumetric mesh is a set of volumetric elements.
- The *outer boundary lines* of two adjacent surface form features (abbr. *OBLF*) (see Fig. 4(b)) are the set of common edges of the polygons between two features, F_0 and F_1.
- The *oriented bounding box* (abbr. *OBB*) is a type of bounding volume used in computational geometry [26]. In this paper, we use the *OBB* in constructing the splitter groups for decomposition.

For an input volumetric mesh M, our decomposition method consists mainly of the following steps:

1. Segment the external surface mesh of M into reasonable regions, use the feature recognition techniques to identify the surface form features, and group the corresponding surface elements together.
2. Find the *OBLF* s of the features. For each *OBLF*, a splitter group is constructed. Separated by these groups, the inner elements are decomposed into disjoint groups, which are attached to the surrounding features.
3. For each pair of adjacent features, we use the graph cut algorithm to cut the splitter groups. For each splitter group, we build an electric field by using the vertices of its corresponding *OBLF* to compute the weights of the edges in the dual graph and obtain the final result.

We will detail steps 2 and 3 in the following section.

4. Decomposition algorithm

In this paper, the input models of the algorithm are volumetric meshes of engineering models. Let M be an original volumetric mesh. We use G, G_{sf}, G_{in}, ∂M to represent the total elements group, the surface elements group, the inner elements group, and the external surface mesh of M, respectively. The goal of this section

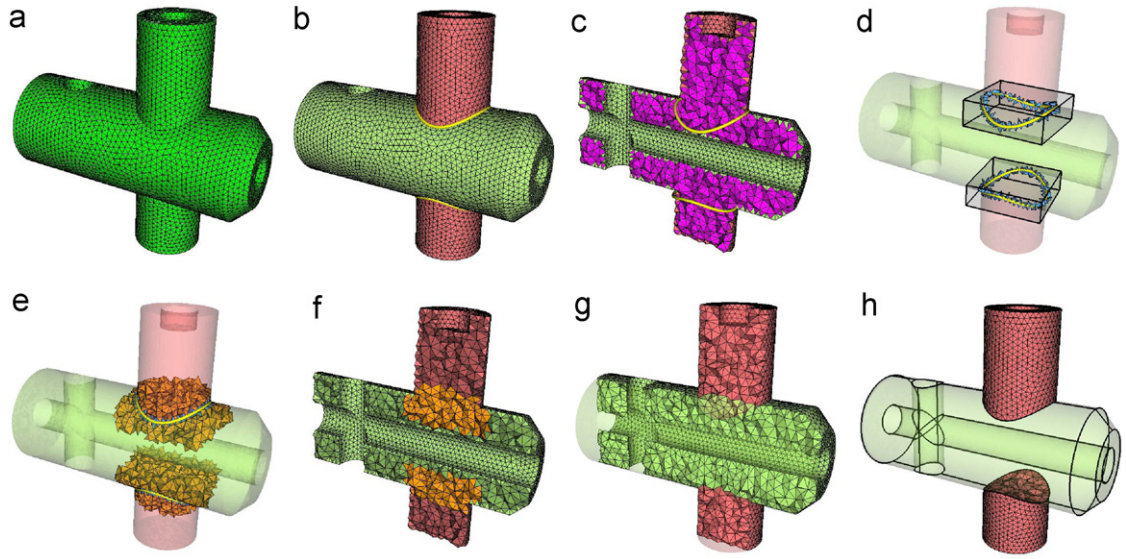

Fig. 5. Decomposition process. (a) is the input tetrahedron mesh, while (b) shows the form features of the surface mesh and their *OBLF* s. The corresponding surface elements of the features and the non-decomposed inner elements are shown in (c), while the surface elements attached to the *OBLF* s and their corresponding bounding boxes are shown in (d). The splitter groups are constructed in (e). Separated by the elements of the splitter groups, the inner elements are decomposed into groups and attached to their corresponding features (f). Finally, using the graph cut algorithm, we decompose the elements of the splitter groups and obtain the results, (g) and (h). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is to decompose G into n semantic groups $\{G_i\}$, such that

$$G = \bigcup_{i=0}^{n-1} G_i, G_i \cap G_j = \emptyset (\forall i \neq j) \quad (1)$$

4.1. Surface decomposition and feature recognition

As mentioned above, our method is an outer-to-inner method. The first step is to decompose ∂M into semantic regions. In this paper, the input model is a finite element model, and the quality of the polygon faces of ∂M is often good. In this case, many segmentation approaches can produce segments well. Furthermore, for engineering models, the boundary of each region of the surface often consist of sharp edges (some adjacent regions may have smooth blend boundaries). Thus, we adopt the method proposed in [27] to decompose the external surface mesh of M in our implementation. It should be pointed out that each quadrilateral on the surface mesh of the hexahedral mesh should be split into two triangles while performing the segmentation. With these surface regions, the adjacent graph of the regions is constructed. Then the graph-based method [27] is employed to recognize the surface form features. In general, a predefined form feature library is built. Because we aim at decomposing the volumetric meshes, the inner side of the form feature should occupy some volume space. Thus, the concave form features, e.g. holes, are attached to the adjacent non-concave form features. In our implementation, we also allow users to interactively define the form features. Because the segmentation of surface meshes and the techniques of feature recognition have been extensively studied, we will cover the details in this paper. Fig. 5(b) shows the surface form features of the tetrahedral mesh in Fig. 5(a).

4.2. Splitter group construction and inner elements partition

Suppose ∂M has been decomposed into a surface feature set $\partial M = \{F_i\}$. It is obvious that every surface polygon face in ∂M has its corresponding surface element in M. After ∂M is decomposed into form feature combinations, the surface elements G_{sf} (as seen in Fig. 5(b)) are grouped. Then, the remaining task is to decompose the inner elements of M. In this step, we use the *OBLF* s to guide the inner decomposition process.

To decompose the inner elements, all *OBLF* s of the adjacent surface features are extracted first. Suppose $l_{i,j} = \{e_k\}$ is the *OBLF* of two features F_i and F_j. Here, e_k is an edge of the surface mesh. As shown in Fig. 5(f), the closer the element $\varepsilon_i \in G_{in}$ is to $l_{i,j}$, the less certain is the group to which the element should belong. Thus, there is a fuzzy elements region near $l_{i,j}$ where the inner elements are likely to be undetermined. Based on this analysis, we use the *OBLF* s to construct the splitter groups in this step.

The algorithm of constructing splitter group $g_{u_{i,j}}$ from $l_{i,j}$ consists of the following steps:

1. Find the surface elements whose edges attach to $l_{i,j}$. We denote these elements as S_l (see elements in blue in Fig. 5(d)).
2. Build an *OBB* [26] O_l to bound S_l.
3. From one element in S_l, collect together the adjacent elements that have at least one vertex inside O_l by applying the breadth-first search algorithm, and add these elements to $g_{u_{i,j}}$.
4. Remove the surface elements from $g_{u_{i,j}}$.

The reason we use *OBB* to help construct the splitter groups is that *OBB* is simple to compute and can surround elements that are near $l_{i,j}$. Figs. 5(d) and (e) show the results of the *OBB* and the splitter groups, respectively.

As illustrated in Fig. 5(f), the splitter groups $G_u = \{g_{u_{i,j}}\}$ are like bars that divide G_{in} into disjoint subgroups $G' = \{g'_m\}$. Combined with G_u, these groups form a partition of G_{in}. We categorize each g'_m into its corresponding adjacent surface feature.

4.3. Decomposition of splitter groups based on electric flux

In the previous steps, the outer and inner elements, except those of splitter groups, are decomposed into corresponding features. To exactly generate the decomposition of G, these splitter groups need to be decomposed.

As described above, a splitter group $g_{u_{i,j}}$ is generated from its corresponding *OBLF*, $l_{i,j}$. As a result, the cutting surface of $g_{u_{i,j}}$ should stay consistent with $l_{i,j}$. A natural idea is to use $l_{i,j}$ as the

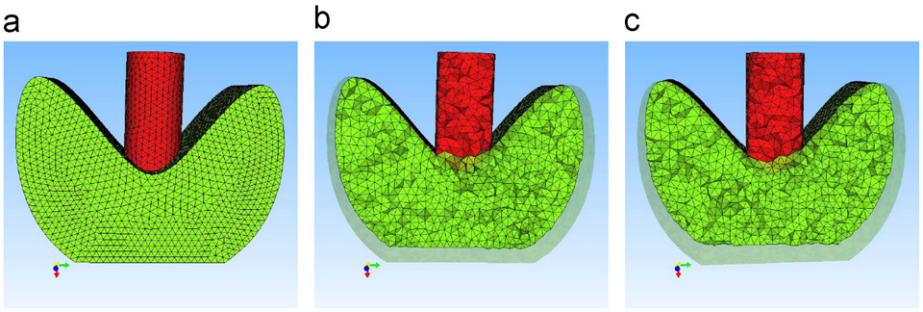

Fig. 6. A model with two features (a). The contact surface of these two features is a saddle surface. Part (b) shows the decomposition result by applying the minimal area rule directly. The result is not very meaningful. Using the minimal flux rule, we can get a better result, shown in (c).

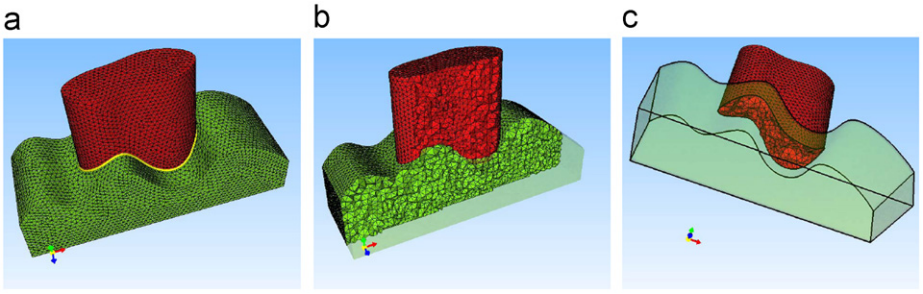

Fig. 7. A model with two features. The base surface of the feature in red is a spline surface (a). Using the proposed algorithm, we get a consistent decomposition with the base surface (see in (b)). Part (c) shows the cutting surface of the two features. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

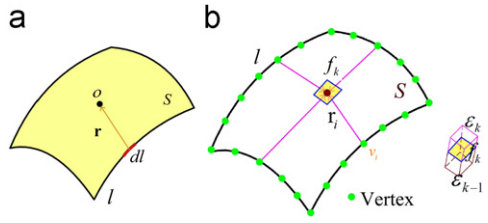

Fig. 8. Electric flux. (a) Flux on a surface. (b) Flux on the contact face elements $\varepsilon_{k-1}, \varepsilon_k$.

boundary to construct a minimal surface for cutting $g_{u_{i,j}}$. However, for a mesh model, $l_{i,j}$ consists of discrete straight lines, and it may be very complex. As shown in Fig. 7, the contact surface of two features is a spline surface. Directly constructing such a minimal surface from the *OBLF* may be very difficult and impracticable. In its discrete form, this solution can be represented as follows: find a cutting surface that divides the splitter group into two subgroups and the sum area of contact faces is minimal. However, as shown in Fig. 6(b), if the minimal area rule is applied directly to the splitter group, the decomposition result is not as good as expected. To avoid this result, we develop the minimal flux rule in this section. We first introduce the electric flux on the volumetric mesh as follows.

Electric flux on volumetric mesh: Suppose there is an electrified curve l in $\mathbb{R}^3$ whose charge is uniformly distributed along this curve. The electric field $\mathbf{E}$ of a point $O \in \mathbb{R}^3$ can then be expressed as

$$\mathbf{E} = \int_l \frac{\rho}{4\pi\varepsilon_0 r^3}\mathbf{r}dl \tag{2}$$

where ρ is the charge density and a constant, and $\mathbf{r}$ is the position vector from dl to O (as shown in Fig. 8). Given a surface S with a boundary l, the electric flux Φ_S of S is

$$\Phi_S = \int_S \mathbf{E}\cdot\mathbf{dS} \tag{3}$$

Apparently, there are many surfaces whose boundaries contain l, and we denote the surface set as $S^{(C)}$. Each surface has its corresponding electric flux Φ. In $S^{(C)}$, there is a bounded surface S_{min}, subject to

$$\Phi_{S_{min}} = \min_{S_i \in S^{(C)}}\{\Phi_{S_i}\} \tag{4}$$

which we regard as the *minimal flux rule*.

As shown in Fig. 8, suppose S is a cutting surface of a volumetric element group; S is then composed of piecewise planes $\{f_k\}$. Here, each $f_k \in S$ is the common face of two elements (see Fig. 8). Let $V = \{v_i\}$ be the vertices of the corresponding *OBLF* of S. In the discretization, $v_i \in V$ is regarded as a unit point electric charge with quantity ε. The discrete form of Eq. (2) can then be written as

$$\mathbf{E} = \sum_{v_i \in V} \frac{\varepsilon}{4\pi\varepsilon_0 r_i^3}\mathbf{r}_i \tag{5}$$

and the flux $\Phi_{f_k} \in S$ of f_k approximates to

$$\Phi_{f_k} = \sum_{v_i \in V} \frac{\varepsilon}{4\pi\varepsilon_0 r_i^3}\mathbf{r}_i \cdot \mathbf{n}_{f_k} A(f_k) \tag{6}$$

where $\mathbf{r}_i$ is the position vector from v_i to the barycenter of f_k, $\mathbf{n}_{f_k}$ is the normal of f_k, and $A(\bullet)$ is the area function that is used to compute the area of f_k. Thus, the discrete form of Eq. (3) is written as

$$\Phi_S = \sum_{f_k \in S} \Phi_{f_k} \tag{7}$$

Now, we apply the minimal flux rule to decompose $g_{u_{i,j}}$. The goal is to find a cutting surface with minimal flux to partition into two reasonable disjoint element groups, g'_i and g'_j. In this step, we

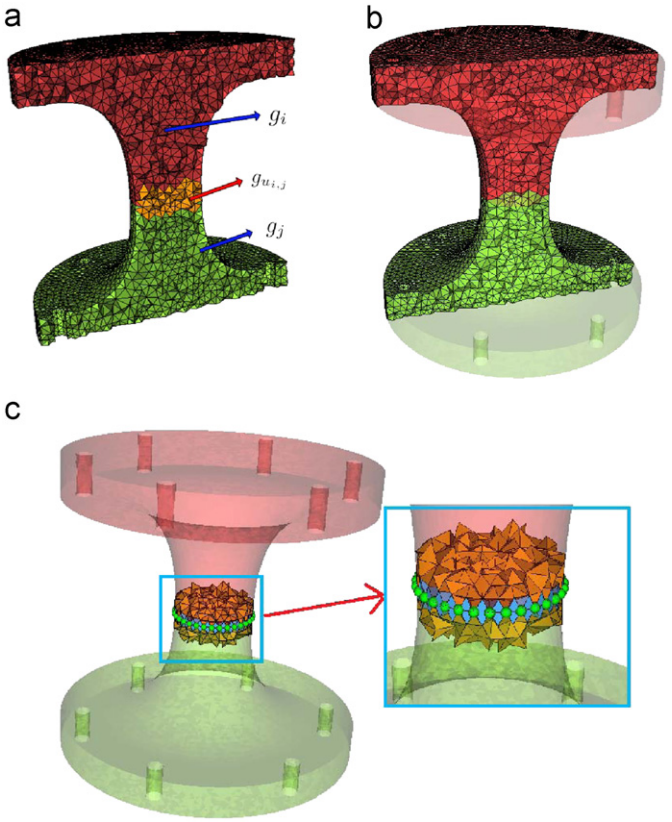

Fig. 9. Decomposition of a model into two components. Part (a) shows the splitter elements groups and (c) shows the electric flux model. Part (b) is the decomposition result.

formulate the cutting problem as a graph partition problem. We first construct the dual graph $D=(N,E)$ of $g_{u_{i,j}}$. In D, the node $n_{\varepsilon_i} \in N$ corresponds to the elements $\varepsilon_i \in g_{u_{i,j}}$, and there is an edge between two nodes if and only if they are face-adjacent. We use the electric flux generated by the vertices of $l_{i,j}$ as the weight of its corresponding edge in D. As a result, this problem turns out to be the minimal cut problem of the network flow [28, chap.26].

It should be noted that two other nodes need to be added to the graph: the source node n_s and the target node n_t. Recall that $g_{u_{i,j}}$ is a splitter group between the two determined groups, g_i and g_j (Fig. 9). If the element $\varepsilon_i \in g_{u_{i,j}}$ is face-adjacent to g_i, then an edge is added between n_{ε_i} and n_s to the dual graph D. The same operation is applied to n_t. In fact, D is an undirect graph. The weight $w(i, j)$ of the edge between two nodes n_{ε_i} and n_{ε_j} is defined as

$$w(i,j)=\begin{cases} |\Phi_{f_k}| & \text{if } n_{\varepsilon_i},n_{\varepsilon_j} \neq n_s,n_t \\ \infty & \text{others} \end{cases} \quad (8)$$

where f_k is the common face of ε_i and ε_j. Here, we ignore the sign of Φ_{f_k} and take its absolute value, because it may be negative, as computed by Eq. (6). Then, by applying the max-flow-min-cut algorithm [28, chap. 26] to D, two separated groups g'_i and g'_j are obtained. Finally, combining these groups with g_i,g_j, we obtain the decomposition result of M. One can see that the graph cut method is similar to the corresponding method for surface polygon mesh segmentation [7,29], but due to differences of source data and aims, certain issues must be resolved. As presented above, the volumetric mesh is more complex, especially the computation of the weights.

From Eqs. (6) and (7), we can see that the minimal flux surface is a weighted minimal area surface. The weight is determined by the electric field and the normal of the face. The

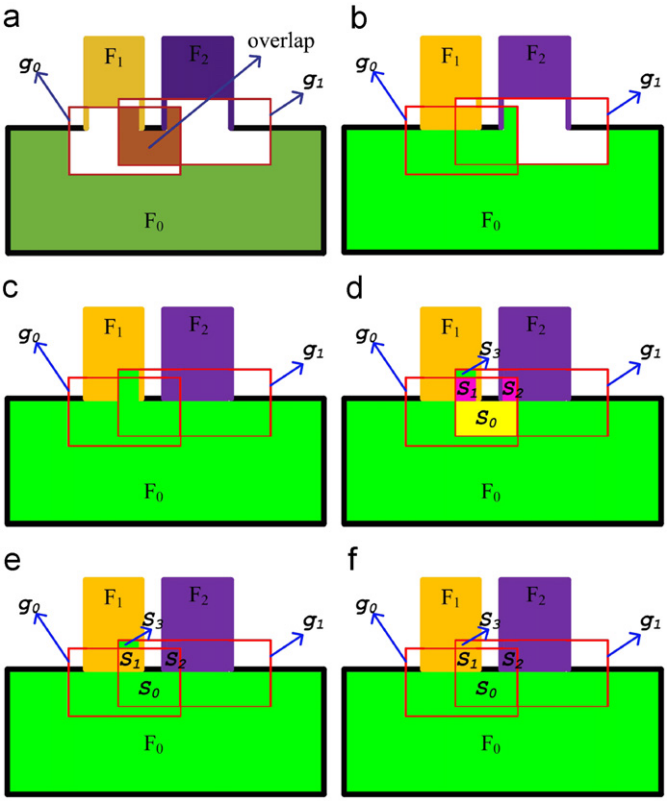

Fig. 10. Postprocessing for overlapping groups. Part (a) shows that g_0 and g_1 overlap. After decomposing g_0, we get (b). When we decompose g_1, we obtain (c). As shown in (d), the subsets S_1 and S_2 of the overlapping parts have two owner features, while S_0 has only one. Check the owner feature of the surrounding surface features of S_1, and assign the elements to F_1. A similar operation is applied to S_2. Part (e) shows that the surrounding component of S_3 is F_1 but not F_0; then, by assigning S_3 to F_1 we obtain the final result, as shown in (f).

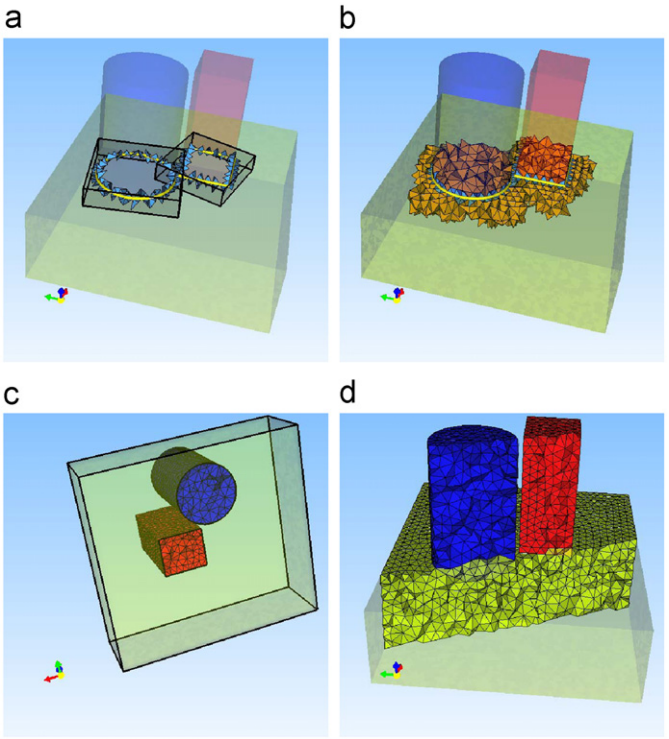

Fig. 11. A model with two close non-adjacent features. (a) Two *OBB* s intersect. (b) The splitter groups overlap. Parts (c) and (d) show the decomposition results.

closer the faces to the *OBLF* s are, the larger the corresponding weights. Moreover, as shown in Fig. 6(c), the cutting surface is more approximate to the saddle surface than the cutting surface of Fig. 6(b) when the minimal flux rule is applied.

4.4. Postprocessing for overlapping groups

As mentioned in Section 4.2, the splitter groups generated by applying the *OBBs* of *OBLF* s may overlap (see Fig. 11) if two non-adjacent features are very close. In this case, these overlapping groups need postprocessing.

In the decomposition of the splitter groups, we first use the minimal flux rule (Section 4.3) to decompose each group. After that, some elements in the overlapping group may be assigned to more than one feature (see S_1 and S_2 in Fig. 10).

Postprocessing for overlapping elements consists of the following steps:

1. Group the elements of the splitter groups into single connected sets by pre-assigned owner features.
2. If the elements of the set have more than one owner features and they are adjacent to one surface feature, then assign this feature as their owner feature.
3. Check the owner features of the surrounding elements for each set. If these features are the same, then assign the feature as the owner feature of the set.

Table 1
List of computation time and the quality of the decomposition results. N_e is the number of elements. N_f represents the number of features. T is the time of decomposition and q_d shows the comparison of the area with the minimal surface.

	N_e	N_f	T (s)	q_d
Fig. 1	90,468	5	4.12	0.87
Fig. 7	106,422	2	18.188	0.91
Fig. 11	50,492	3	0.400	0.95
Fig. 12(a)	44,764	3	7.344	0.85
Fig. 12(b)	51,459	3	2.094	0.78
Fig. 13	44,216	15	0.805	0.93
Fig. 14(a)	45,718	2	5.857	0.88
Fig. 14(c)	45,718	5	2.703	0.80
Fig. 15(a)	35,004	4	0.531	0.87
Fig. 15(c)	5036	4	0.049	0.82
Fig. 16	161,096	7	3.655	0.86
Fig. 17	47,880	4	1.090	0.81
Fig. 19	33,000	7	0.328	1
Fig. 20	98,254	25	1.097	1

5. Implementation and results

We implement our algorithm using Visual C++ 2008 with OpenGL. Because the topological relationship of the volumetric

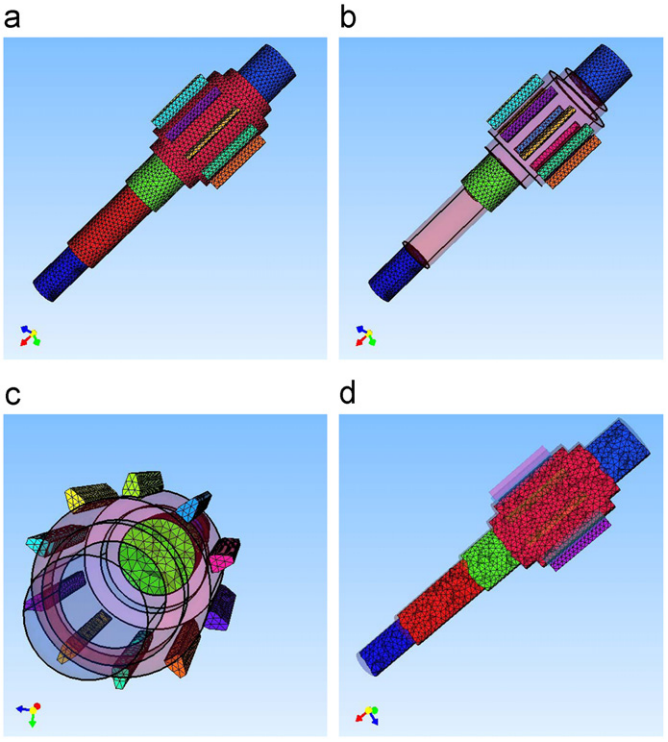

Fig. 13. A bearing model with 15 features.

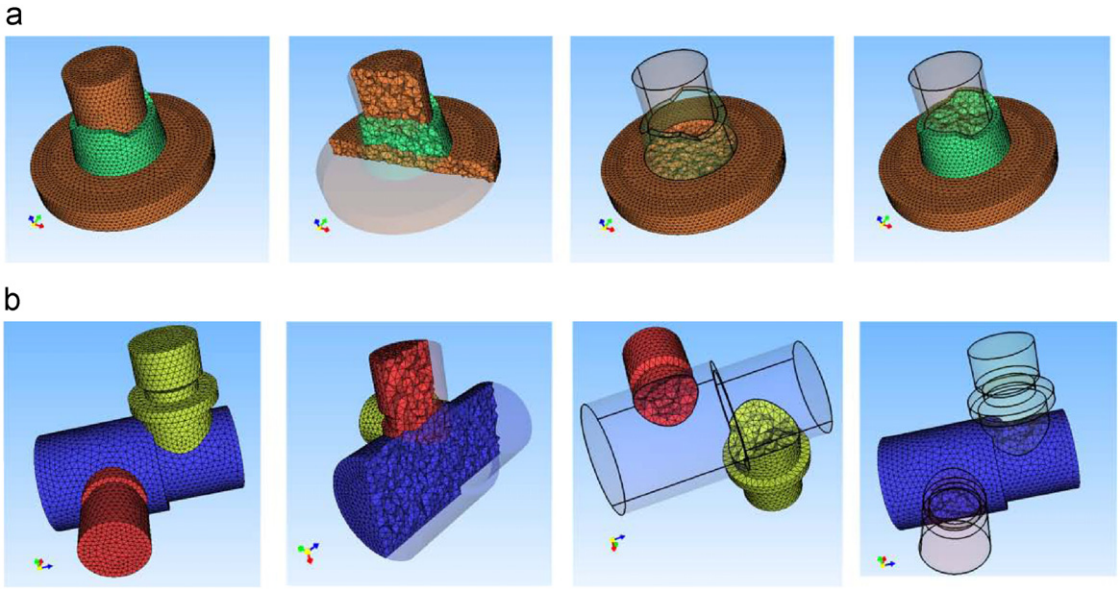

Fig. 12. Examples of our method for various tetrahedral mesh models. The first column shows the surface form features of the volumetric mesh models, while the second column shows the cross-sections of the decomposition results. The third and the fourth columns show the volumetric features of the results.

elements is more complex than that of the polygon mesh, we extend the *Compact Half-face* [30] data structure to represent the volumetric mesh, to support the decomposition process. Our data structure constructs the linking information between different dimensions (0D, 1D, 2D and 3D) and it allows us to access information without traversing the mesh.

We tested our method on many volumetric models, and the results were satisfactory. In Table 1, we show the running time and additional data for our method. The test were performed on a PC with Core 2 Dual 2.4 GHz and 4.0 GB RAM in a single thread.

We compare the areas of the cutting surface with that of the minimal surface of the *OBLF* s. Suppose a mesh contains n cutting surfaces after decomposition; A_i is the area of the cutting surface of two features, and A_i^m is the area of the corresponding *OBLF*. We compute the closeness to the minimal surface q_d as follows:

$$q_d = 1 - \frac{1}{n}\sum_{i=0}^{n-1}\frac{A_i - A_i^m}{A_i^m} \tag{9}$$

Because it is difficult to build an exact minimal surface of a closed curve, we first triangulate the closed curve and then use the areas of the triangles to approximate A_i^m in our implementation. Numeric data are listed in Table 1.

Fig. 12 shows two examples with different intersections of the features. In Fig. 12(a), the green component has two neighbors and two *OBLF* s. One *OBLF* lies on a plane, and the other does not. Both cases are decomposed well. The form features of Fig. 12(b) are defined by the user. Fig. 13 shows the decomposition of the bearing model with 15 features. We can see that our method can decompose such a complex model well.

The decomposition result of a turbo model is shown in Fig. 1. The base surface of the sheet features is a spline surface in the original model. Our method can decompose the model well and can generate a consistent decomposition with the boundaries. Another similar model is shown in Fig. 7.

Fig. 14 shows an example of decomposing a mesh in different ways. As shown in Fig. 14, the *OBLF* s of two features in (a) lie on the same plane while those in (c) do not. In the second case, it is difficult to construct a surface that can directly cut the features. However, by using the proposed method, good decomposition results for both cases are obtained.

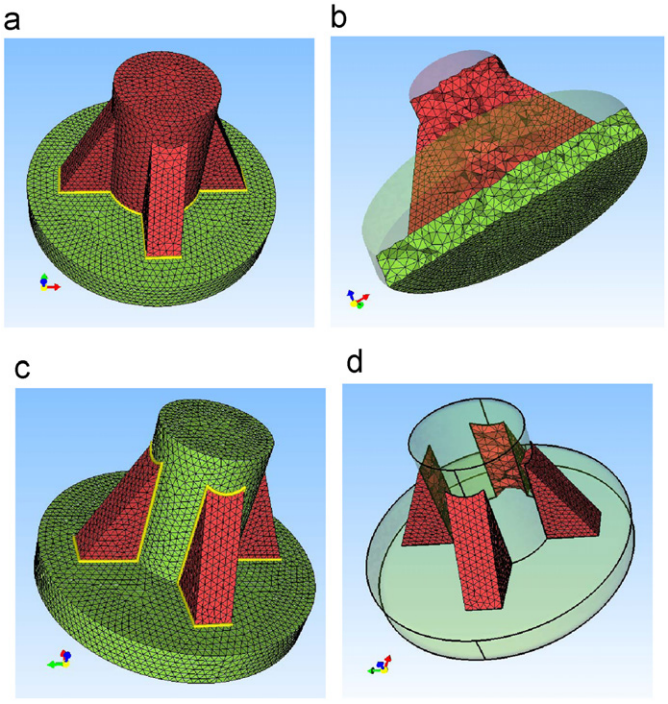

Fig. 14. Using different ways to decompose a model.

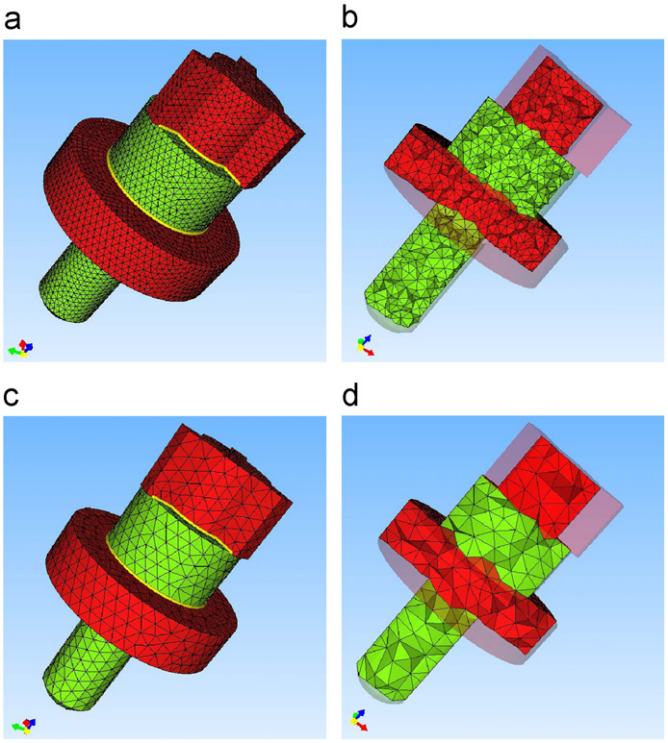

Fig. 15. Decomposing a model with different size of elements.

Our method can generate similar decomposition results with different sizes of the elements. In Fig. 15, the size of the elements in (c) is two times the size of the elements in (a). The data in Table 1 demonstrate that the decomposition results are very similar, although their sizes are different.

Fig. 16 shows a decomposition result of a complex model with seven features. The two surface form features that contact the green feature are selected manually. Then, applying the proposed method, we decompose the model into seven components.

Figs. 19 and 20 show some examples of hexahedral meshes. In Fig. 20, the hexahedral gear model contains 25 features and has been successfully decomposed. From these figures, it can be seen that if the elements are attached to the contact surfaces between the form features, our method can generate consistent decomposition with these surfaces.

A model with two intersected cylinders is shown in Fig. 17. In this model, four surface components meet at a single point. Thus, four splitter groups overlap. Using the postprocessing described in Section 4.4, we get the result in Fig. 17(d).

5.1. Discussion and limitations

Currently, we are using the method presented in [27] to segment the surface mesh and to recognize the surface form features. It seems that the proposed method cannot always generate the most reasonable decomposition result. As shown in Fig. 5, the model may have another decomposition result, which would be that the intersection region of the cylinders belongs to the form feature in red. This result happens because the rationality of the decomposition results depends on the feature recognition method we used. To make our approach more practical, the feature recognition method needs to be improved.

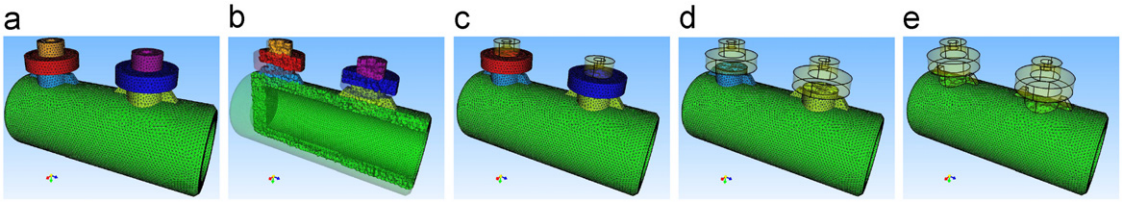

Fig. 16. A complex tetrahedral mesh model with seven features.

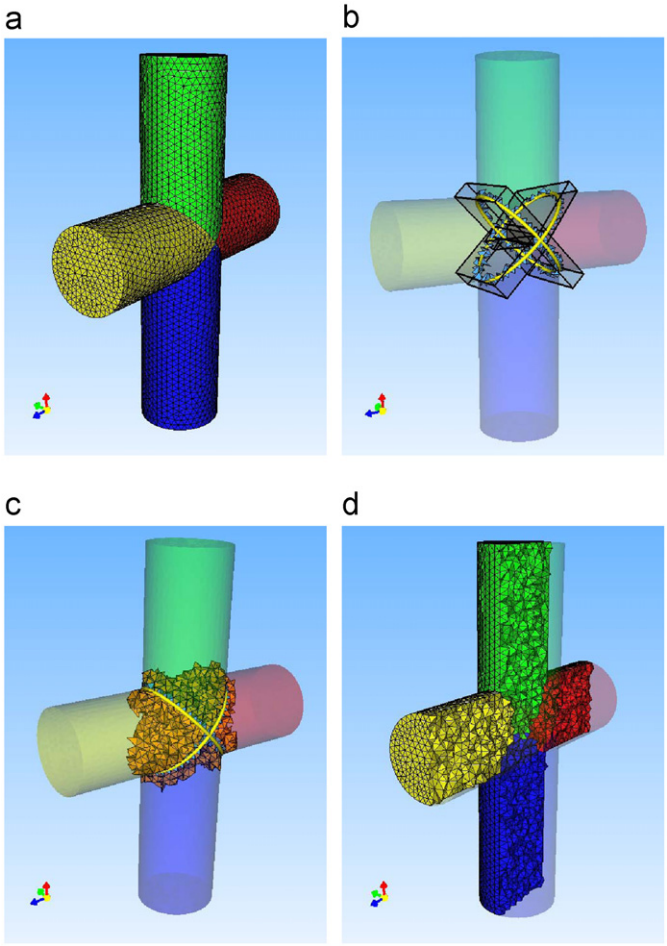

Fig. 17. The decomposition of the intersect on cylinder model. (a) The model with four form features. Part (b) shows four *OBB* s and (c) shows the corresponding splitter groups. Part (d) is the result.

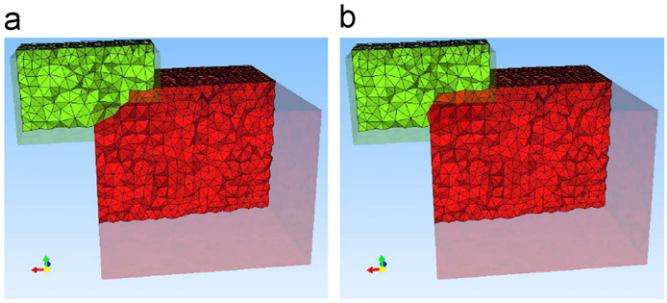

Fig. 18. (a) The decomposition result using our method. (b) Another possible decomposition result.

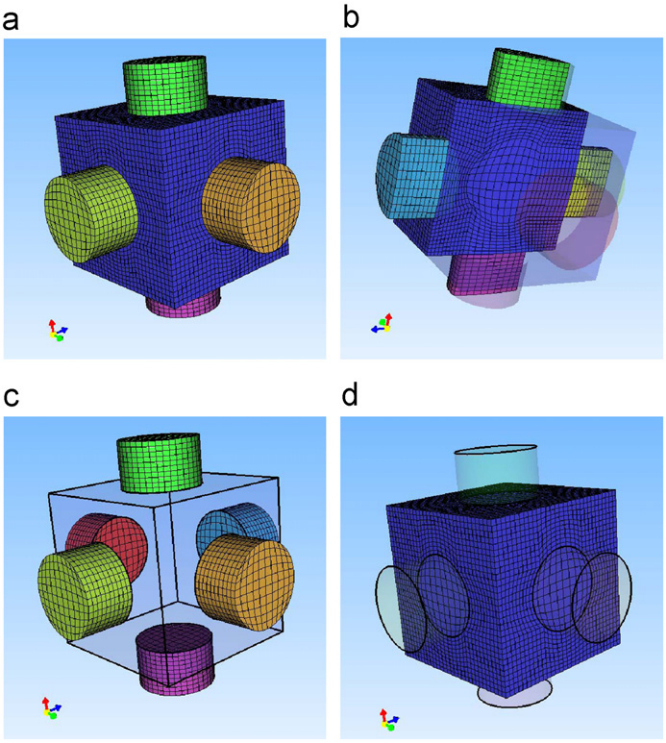

Fig. 19. The decomposition of a hexahedral mesh model with seven features. This model is generated by CUBIT.

For a given surface feature, there may be various decomposition results. For example, Fig. 18 shows two decomposition results of the given surface feature in green. Our method can generate the decomposition result that the weighted area of the splitting surface is minimal, but the most reasonable result cannot always be guaranteed. Perhaps domain knowledge could be incorporated into our method for a better result.

6. Conclusions

In this paper, we have presented an effective approach for the automated decomposition of a volumetric mesh. In our approach, a tetrahedral or a hexahedral mesh model can be automatically decomposed into semantic components by first segmenting the outer surface mesh into semantic features and then obtaining the semantic features, consisting of volumetric elements, by employing the graph cut algorithm. The characteristics of the proposed approach include:

1. The quality of the decomposed semantic features, consisting of volumetric elements, is guaranteed by using our graph cut algorithm.

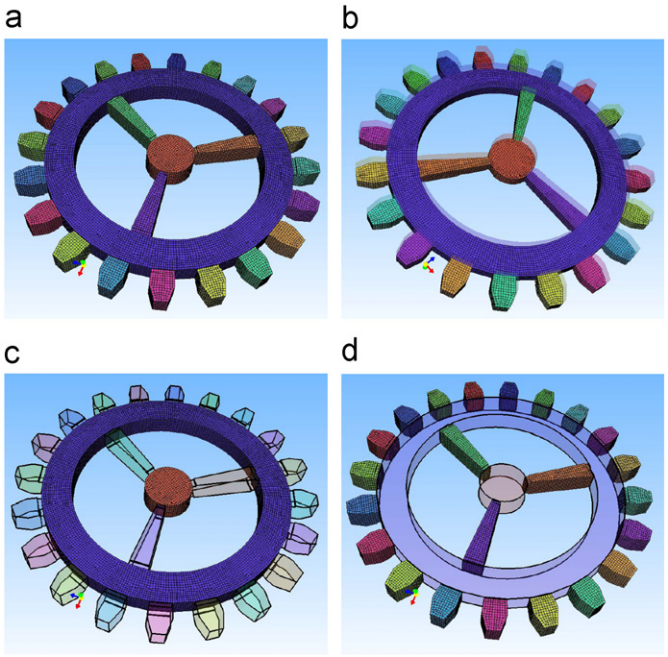

Fig. 20. The decomposition of the gear model. (a) The model has 25 features. Part (b) shows the cross-section of the decomposition.

2. The decomposed semantic features can be complex predefined features with complicated boundary surfaces and curves.
3. The method is quite efficient and can handle both tetrahedral and hexahedral meshes.

Compared to the method in [24], our method begins the decomposition with the surface mesh and feature recognition because the form features are well-defined in engineering models and these form features contain important semantic meanings in the engineering field. The proposed method also considers the boundaries of the adjacent features, which makes the decomposition result consistent with the boundaries.

As discussed in Section 5.1, the rationality of the decomposition depends on the form features. We will improve the adopted feature recognition method so as to make our approach more practical. Another possibility for future work is to study the validation of the decomposition of the volumetric mesh with complex form features.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and their help to revise our paper. This paper is supported by NSFC (No. 60736019).

References

[1] Varady T, Martin R, Cox J. Reverse engineering of geometric models—an introduction. Computer-Aided Design 1997;29(4):255–68.
[2] Zuckerberger E, Tal A, Shlafman S. Polyhedral surface decomposition with applications* 1. Computers & Graphics 2002;26(5):733–43.
[3] Attene M, Falcidieno B, Spagnuolo M. Hierarchical mesh segmentation based on fitting primitives. The Visual Computer 2006;22(3):181–93.
[4] Funkhouser T, Kazhdan M, Shilane P, Min P, Kiefer W, Tal A, et al. Modeling by example. ACM Transaction on Graphics 2004;23(3):652–63.
[5] Karni Z, Gotsman C. Spectral compression of mesh geometry. In: Proceedings of SIGGRAPH. ACM; 2000. p. 279–86.
[6] Li X, Woon T, Tan T, Huang Z. Decomposing polygon meshes for interactive applications. In: Proceedings of the 2001 symposium on interactive 3D graphics. ACM; 2001. p. 35–42.
[7] Katz S, Tal A. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Transactions on Graphics (TOG) 2003;22(3):954–61.
[8] Attene M, Katz S, Mortara M, Patane G, Spagnuolo M, Tal A, et al. Mesh segmentation—a comparative study. In: IEEE international conference on shape modeling and applications; 2006. p. 7.
[9] Shamir A. A survey on mesh segmentation techniques. Computer Graphics Forum 2008;27(6):1539–56.
[10] Lou R, Pernot J, Mikchevitch A, Véron P. Merging enriched finite element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance. Computer-Aided Design 2010;42(8):670–81.
[11] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. ACM Transaction on Graphics (TOG) 2009;28(3).
[12] Mangan A, Whitaker R. Partitioning 3D surface meshes using watershed segmentation. IEEE Transactions on Visualization and Computer Graphics 1999;5(4):308–21.
[13] Katz S, Leifman G, Tal A. Mesh segmentation using feature point and core extraction. The Visual Computer 2005;21(8):649–58.
[14] Shlafman S, Tal A, Katz S. Metamorphosis of polyhedral surfaces using decomposition. Computer Graphics Forum 2002;21(3):219–28.
[15] Yamauchi H, Lee S, Lee Y, Ohtake Y, Belyaev A, Seidel H. Feature sensitive mesh segmentation with mean shift. In: 2005 international conference shape modeling and applications; 2005. p. 236–43.
[16] Reniers D, Telea A. Skeleton-based hierarchical shape segmentation. In: Shape modeling and applications SMI'07. IEEE international conference on IEEE; 2007. p. 179–88. [ISBN 0769528155].
[17] Lai Y, Hu S, Martin R, Rosin P. Rapid and effective segmentation of 3D models using random walks. Computer Aided Geometric Design 2009;26(6):665–679.
[18] Skraba P, Ovsjanikov M, Chazal F, Guibas L. Persistence-based segmentation of deformable shapes. In: Computer vision and pattern recognition workshops (CVPRW), 2010 IEEE computer society conference on IEEE; 2010. p. 45–52.
[19] Zheng Y, Tai C. Mesh decomposition with cross-boundary brushes. Computer Graphics Forum 2010;29(2):527–35.
[20] Lavoué G, Dupont F, Baskurt A. A new CAD mesh segmentation method based on curvature tensor analysis. Computer-Aided Design 2005;37(10):975–87.
[21] Hwan Kim D, Dong Yun I, Uk Lee S. Boundary-trimmed 3D triangular mesh segmentation based on iterative merging strategy. Pattern Recognition 2006;39(5):827–38.
[22] Várady T, Facello M, Terék Z. Automatic extraction of surface structures in digital shape reconstruction. Computer Aided Design 2007;39(5):379–88.
[23] Agathos A, Pratikakis I, Perantonis S, Sapidis N, Azariadis P. 3D mesh segmentation methodologies for CAD applications. Computer-Aided Design and Applications 2007;4(6):827–41.
[24] Attene M, Mortara M, Spagnuolo M, Falcidieno B. Hierarchical convex approximation of 3D shapes for fast region selection. Computer Graphics Forum 2008;27(5):1323–32.
[25] Masuda H, Yoshioka Y, Furukawa Y. Preserving form features in interactive mesh deformation. Computer-Aided Design 2007;39(5):361–8.
[26] Gottschalk S, Lin M, Manocha D, Tree O. A hierarchical structure for rapid interference detection. In: Proc ACM Siggraph, vol. 96; 1996. p. 171–80.
[27] Gao S, Zhao W, Lin H, Yang F, Chen X. Feature suppression based CAD mesh model simplification. Computer Aided Design 2010;42(12):1178–88.
[28] Thomas H, Cormen CE. Introduction to algorithms. The MIT press; 2001.
[29] Golovinskiy A, Funkhouser T. Randomized cuts for 3D mesh analysis. ACM Transactions on Graphics 2008;27(5):145.
[30] Lage M, Lewiner T, Lopes H, Velho L. CHF: a scalable topological data structure for tetrahedral meshes. In: Proceedings of the XVIII Brazilian symposium on computer graphics and image processing 2005; 2005. p. 349–56.