Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2012
August 12-15, 2012, Chicago,IL, USA

DETC2012-70856

HEXAHEDRAL MESH CUTTING USING GEOMETRIC MODEL WITH NEW BOUNDARIES WELL MATCHED

Hua Zhu
State Key Lab of CAD & CG
Zhejiang University,
Hangzhou 310027, China

Shuming Gao[*]
State Key Lab of CAD & CG
Zhejiang University,
Hangzhou 310027, China

Chuhua Xian
State Key Lab of CAD & CG
Zhejiang University,
Hangzhou 310027, China

ABSTRACT

Hexahedral mesh generation is difficult and time-consuming. To avoid the complicated hexahedral mesh regeneration after each variational design, hexahedral mesh editing can be used. In this paper, an accurate hexahedral mesh cutting approach using geometric model is proposed, and the part of the geometric model inside the mesh model can be complex and arbitrary. In the approach, all the newly added geometric entities resulted from mesh cutting are first generated by performing the subtraction operation between the mesh model and the geometric model. Then, for each newly added geometric element, including point, edge and face, the mesh nodes that need to be moved onto it, are determined and repositioned with the mesh quality considered. To ensure the rationality of mesh nodes determination, for each newly added edge, the mesh nodes are identified using shortest path algorithm. Finally, the mesh elements that should not be in the resultant mesh model are deleted, Pillowing and Smoothing operations are further conducted to improve the mesh quality. Some preliminary results are given to show the feasibility of the approach.

INTRODUCTION

Finite Element Method (FEM) has been widely used to test the validation of product models all these decades. By using this method, the geometric model constructed by engineers need to be discretized to standard finite elements for solving. Among various kinds of finite elements, hexahedron element is the most important one, because it can achieve higher analysis accuracy with fewer mesh elements. There are several approaches to hexahedral mesh generation: submapping, sweeping, whisker-weaving, plastering, and grid-based method. Among these methods, submapping and sweeping methods can be used to generate high quality hexahedral mesh model, but manual model decomposition is usually needed for most product models. For other methods, most of the hexahedral mesh models can be generated automatically using them, however, the mesh quality cannot be ensured. In a word, automatic generation of high quality all-hexahedral mesh is still a challenging problem.

Product design is usually carried out in an iterative way. After each variational design, Finite Element Analysis needs to be conducted again typically. Traditionally, the mesh model is re-generated to support the required re-analysis. Since mesh re-generation is very time-consuming and difficult, especially for hexahedral mesh, two types of methods are adopted to solve this problem: remeshing based on mesh re-use and direct mesh editing. In view that the design modifications are usually minor and most parts of the mesh model can be re-used, the first type of methods saves the time by performing the remeshing only for the modification zone instead of the whole modified model. In recent years, direct mesh editing technique has been paid more and more attention, because it is more efficient without the need to modify the CAD model first. But existing mesh editing methods can not support the precise and complex mesh editing operations well yet.

Although direct editing on hexahedral mesh is very useful, there is little research on hexahedral mesh editing so far because

[*]Corresponding author E-mail address: smgao@cad.zju.edu.cn.

of the complexity of handling hexahedral mesh. Different from tetrahedral mesh, hexahedral mesh with high quality is highly structured. In order to effectively support the handling of hexahedral mesh, dual mesh is put forward to represent the topology structure of hexahedral mesh, and various operations for the topological modification of hexahedral mesh are developed. Since the boundary of the mesh model can not be changed by these operations, mesh cutting and mesh union are imperative. However, the existing methods of hexahedral mesh cutting are still very preliminary: the geometric model used for cutting is simple, and the new boundaries resulted from cutting are not well matched.

In this paper, we propose an approach to hexahedral mesh cutting. The goal of the approach is to ensure that the resultant hexahedral mesh well matches all the new boundary entities resulted from the cutting with complex geometric model.

RELATED WORK

For triangle mesh model, various methods have been studied on mesh editing. Especially, there are methods on mesh editing based on CAD operations, but only some specific operations are applied. Liu et al. [1] presented an approach to blending triangle mesh model to smooth the connecting between two mesh models. Virtual blending surface is constructed for the blending region and vertices are moved onto it, and the radius of the rolling ball can be controlled by users which is efficient. Mesh blending is presented for visualization, and Lou et al. [2] presented a method merging Finite Element mesh models. The intersection lines between two mesh models are determined and ensured in the process of merging, then nearby triangles are regenerated. However, these mesh models have to be generated before merging, and we [3] developed a mesh editing method to extrude faces on triangle model directly, then merge their mesh model with the original mesh model.

Besides triangle mesh models, there are a few studies on tetrahedral mesh editing, and most of them only focus on mesh deformation but not merging or cutting. The techniques for triangle mesh deformation are mature, so tetrahedral mesh models are usually deformed according to the deformation of their surface mesh. The mapping between nodes on the surface of the mesh model and nodes at the internal of the mesh model is established, and the internal mesh nodes are repositioned according to the repositioning of the nodes on the surface. Mollemans et al. [4] deform tetrahedral mesh models according to the simulation of spring model. Song [5] proposed a uniform non-linear optimization solution to manipulate the deformation of tetrahedral mesh model. However, the connectivity between mesh nodes are not considered which will result in distorted tetrahedrons. On the other hand, the boundary topology of the mesh model is not varied after deformation. Moreover, if large deformation is conducted, the mesh model has to be optimized for the number of nodes is not varied after deformation. Lou et al. [6] presented a approach to drilling holes on tetrahedral mesh model, and group information of FE mesh model can be kept in the result mesh model.

Editing on hexahedral mesh model is closely related to hexahedral mesh generation, and it is still not fully resolved. Mesh cutting [7] is proposed to generate complex hexahedral mesh model using mesh cutting on already generated mesh model. In their method, mesh nodes are projected onto the face of the geometric model. However, edges and vertexes of the geometric model are not considered, and the boundary constraint may be not ensured.

OVERVIEW OF THE APPROACH

In order to achieve powerful and accurate hexahedral mesh cutting operation, a new hexahedral mesh cutting approach is proposed in this paper. The input of the approach is a given hexahedral mesh model and a geometric model used to cut the hexahedral mesh model, and the output of the approach is a valid hexahedral mesh model taking the related geometric entities of the given geometric model as its boundary constraints. Unlike the existing methods that only get the resultant hexahedral mesh model matched to the related faces of the geometric model used for cutting, all the newly added geometric entities resulted from the hexahedral mesh cutting are used in the approach for matching, including all the newly added points and edges. By doing this, the accuracy of the resultant hexahedral mesh model is guaranteed as much as possible. In the approach, after all the newly added geometric entities are generated, the hexahedral mesh model is matched to all the new added points first, then all the newly added edges, and finally all the newly added faces. In order to effectively determine the mesh edge elements that need to be matched to a newly added edge, A-Star algorithm is adopted. In addition, the approach allows the geometric model used for cutting to be arbitrary complex.

As illustrated by Fig. 1, our approach to hexahedral mesh cutting consists of the following four steps:

1. Generating all the newly added geometric entities;
2. Matching the related mesh nodes to each newly added geometric entity;
3. Removing the hexahedrons inside the geometric model;
4. Smoothing the resultant hexahedral mesh model.

GENERATION OF NEWLY ADDED GEOMETRY ENTITIES

In order to perform hexahedral mesh cutting using geometric model, the newly added geometry entities after subtracting mesh model from geometric model is analyzed and generated in this section. If simply subtract the mesh model from the geomet-

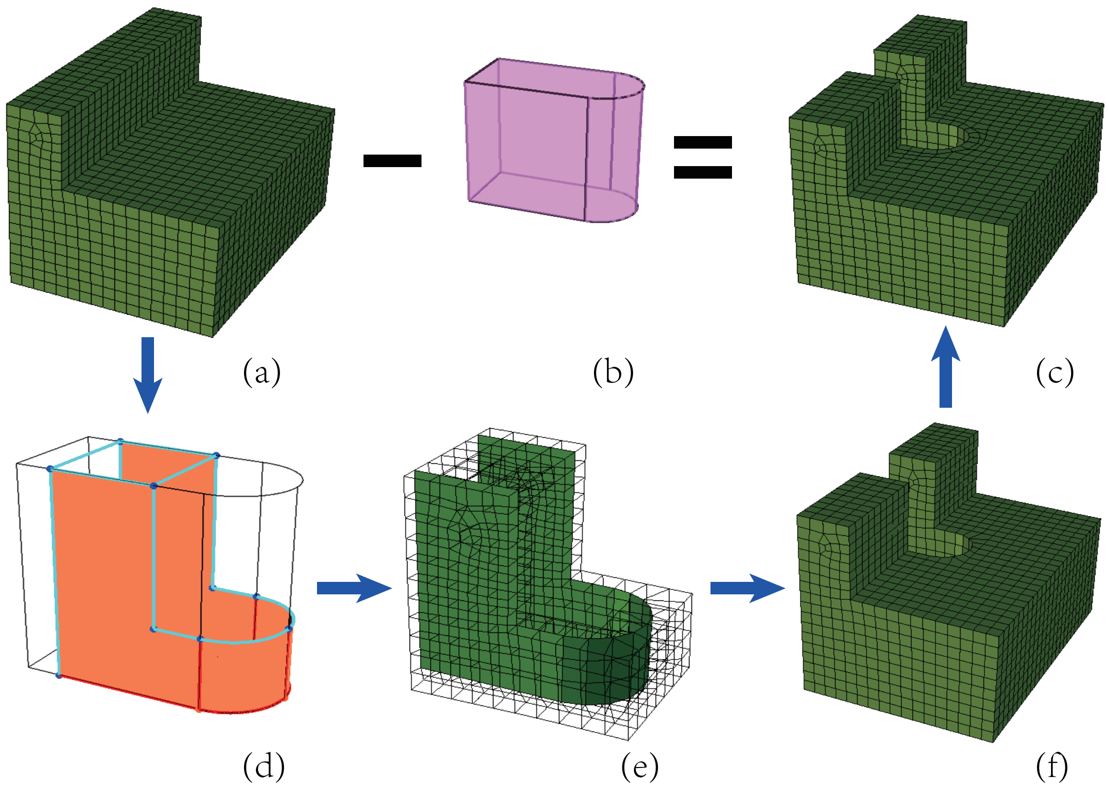

FIGURE 1. Hexahedral mesh cutting using geometric model.(a)input mesh model, (b)geometric model for cutting, (c)resultant mesh model, (d)newly added geometry entities, (e)nodes matched on newly added geometry entities, and (f)removing of hexahedrons inside the geometric model.

ric model, the result mesh model which has incomplete hexahedrons is not a valid mesh model. After this simple subtraction, we use *newly added geometry entity set* to represent the newly added geometry entities of the resultant model, and it is denoted as NAGES for short. These geometry entities include points, edges and faces, and their generation are explained later.

Newly Added Geometry Entities

From the above definition of NAGES, we find that the geometry entities of them are parts of the geometry model inside or at the surface of the mesh model. As shown in Fig. 2(a), the mesh model is intersected with the geometric model, and in Fig. 2(c), NAGES is consist of face GHLK and face EILH, together with their edges and points. However, some of the geometry entities at the surface of the mesh model are not used for mesh cutting, so they are not included in NAGES, e.g. face EFGH, EIJF and FJKG, together with their edges and points.

Therefore, NAGES can be computed using elements of the mesh model and entities of the geometric model. For each mesh element and each geometry entity, there are four types of relationship between them:

1. The mesh element is coincident with the geometry entity, and they are in the same dimension, e.g. a mesh edge element coincides with a geometry edge.
2. The mesh element is inside the geometry entity, and is at a dimension not higher than the geometry entity, e.g. a mesh edge element lies on a geometry face.
3. Part of the geometry entity is inside the mesh element, and the mesh element is in a dimension not lower than the geometry entity, e.g. a geometry edge lies on a mesh face element.
4. The intersection part of the mesh element and the geometry entity is at a lower dimension than both of them, e.g. a mesh face element and a geometry edge are intersected at a point.

For relationships 3 and 4, the mesh elements are on the surface of the mesh model in our method. Then we denote the mesh edge element on the surface of the mesh model as *boundary edge element*, and the mesh face element on the surface of the mesh model is denoted as *boundary face element*. Under the above situations, mesh elements and geometry entities are intersected, and the result can be points, edges and faces. For convenient, the set of points in NAGES is denoted as NAGESp, the set of edges in NAGES is denoted as NAGESe, and the set of faces in

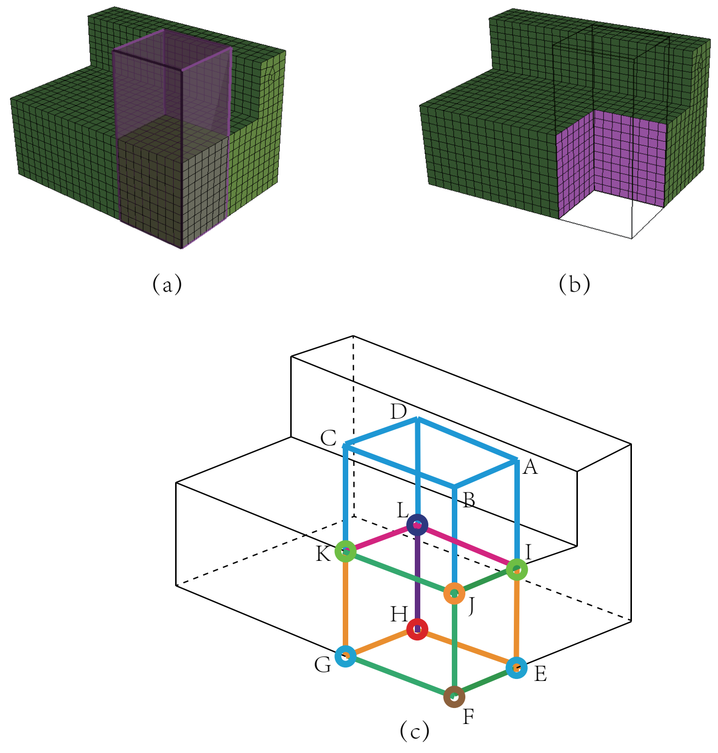

FIGURE 2. Newly added geometry entities: (a)mesh model and geometric model, (b)newly added geometric entities in purple and (c)types of newly added point, edge and face.

NAGES is denoted as NAGESf. Additionally, for relationships 3 and 4, the intersection parts consist of intersection points and intersection lines. Therefore, NAGESp and NAGESe are divided into two types individually as follows:

The set of points in NAGESp that are geometric vertexes inside or on the surface of the mesh model, and retained in the result mesh model. It is denoted as NAGESgp, e.g. point E, G and H in Fig. 2.

The set of points in NAGESp that are intersection points between the geometric model and the surface of the mesh model, and retained in the result mesh model. It is denoted as NAGESip, e.g. point I, K and L in Fig. 2.

The set of edges in NAGESe that are sections of geometric edges, these edges are inside or on the surface of the mesh model, and retained in the result mesh model. It is denoted as NAGESge, e.g. line EI, HE, GH, KG and LH in Fig. 2.

The set of edges in NAGESe that are intersection lines between the geometric model and the surface of the mesh model, and retained in the result mesh model. It is denoted as NAGESil. e.g. line IL and LK in Fig. 2.

Additionally, NAGESf is the set of the parts of geometric faces that are inside the mesh model. And for faces on the surface of the mesh model, e.g. face EFGH, EIJF and FJKG in Fig. 2(c), the mesh face elements coincident with them are to be removed or there is no need to perform node repositioning.

In the rest part of this section, NAGES is generated from NAGESp, NAGESe to NAGESf, and the procedures are shown in Fig. 3. After the generation of NAGESf, if points of NAGESp and edges of NAGESe are without faces of NAGESf associated, they are removed finally.

Generation of NAGESp

NAGESp consists of NAGESgp and NAGESip, they are generated individually. NAGESgp is the set of vertexes of geometric model inside or on the surface of the mesh model. NAGESip is the set of intersection points between the boundary of the geometric model and the mesh model, and it is generated according to the following situations:

If a mesh node on the surface of the mesh model is on the geometric edge, this mesh node belongs to NAGESip, e.g. point A in Fig. 3(a).

If a mesh edge element on the surface of the mesh model is intersected with the geometric edge, this intersection point belongs to NAGESip, e.g. point B in Fig. 3(a).

If a mesh face element on the surface of the mesh model is intersected with the geometric edge, this intersection point belongs to NAGESip, e.g. point C in Fig. 3(a).

If a mesh node on the surface of the mesh model is on the geometric face, this mesh node belongs to NAGESip.

If a mesh edge element on the surface of the mesh model is intersected with the geometric face, this intersection point belongs to NAGESip, e.g. point D in Fig. 3(a).

As shown in Fig. 3(a), the geometric model is intersected with the mesh model, the points of NAGESgp are the red points, and the points of NAGESip are in blue. Only hexahedrons intersected with and inside the geometric model are shown in the figure, and the green mesh face elements are boundary face elements of the mesh model.

Generation of NAGESe

See the result of NAGESe generation shown in Fig. 3(b), the red lines are edges of NAGESge, and the cyan lines are intersection lines of NAGESil. For NAGESge, it is the result of the intersection between geometric edges and the mesh model, so the parts of geometric edges at the internal or on the surface of the mesh model are determined as members of NAGESge. For example, there is a geometric edge *MN* as shown in Fig. 3(b), then line *PN* which is the section of *MN* inside the mesh model is determined as a member of NAGESge.

As explained before, NAGESil is the set of intersection lines between geometric faces and the surface of the mesh model. In order to construct intersection lines, the key problem is how to connect intersection points of NGAESip on the surface of

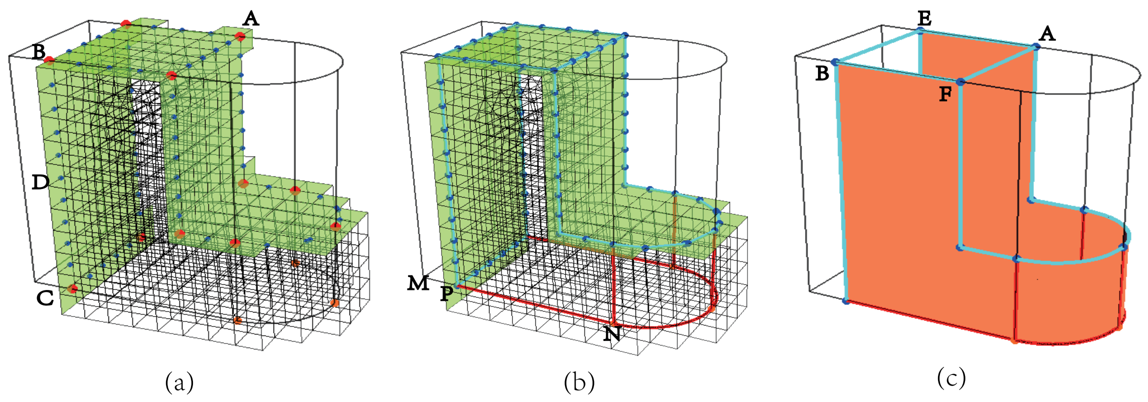

FIGURE 3. Generation of NAGES: (a)generation of NAGESp, (b)generation of NAGESe, and (c)generation of NAGESf.

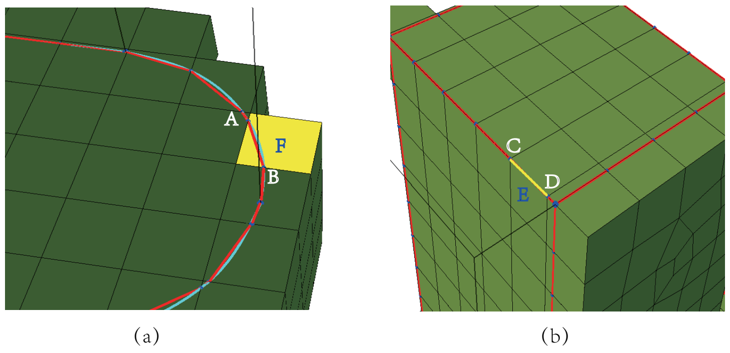

FIGURE 4. Types of mesh element between two intersection points: (a)mesh face element, and (b)mesh edge element.

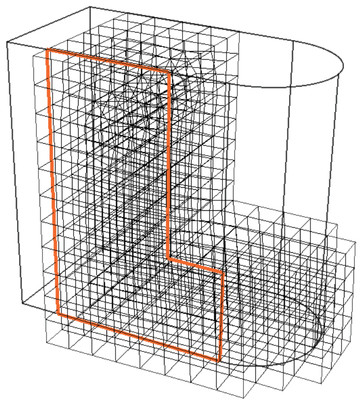

FIGURE 5. Edge loop of NAGESf.

the mesh model. The geometric face may be intersected with a boundary edge element at the end node or the internal of it. Therefore, for an intersection point, the situations for the mesh element between two intersection points are shown in Fig. 4:

1. There is a boundary face element between two intersection points intersected with the geometric face. As shown in Fig. 4(a), there is a mesh face element F between the two intersection points A and B;
2. There is a boundary edge element between two intersection points intersected with the geometric face. As shown in Fig. 4(b), there is a mesh edge element E between the two intersection points C and D.

Using the above connection information, two intersection points are connected if they are adjacent to the same mesh element. Therefore, an intersection line can be constructed by connecting intersection points of NAGESip. And these intersection lines are cyan lines in Fig. 3(b).

Generation of NAGESf

In order to generate NAGESf, the geometric faces are intersected with the mesh elements to determine the parts of the faces inside the mesh model. And there are two situations: the geometric faces are coincident with the mesh face elements, and they are intersected with the hexahedrons. For convenient, the first situation can be seen as a special case of the second one, so it is not discussed here. As NAGESe is generated, NAGESf can be seen as the parts of geometric faces segmented using NAGESe. In Fig. 5, the side face of the geometric model is segmented by the orange loop of intersection lines and a section of the geometric edge. Then the part of the geometric face inside the edge loop is determined as a member of NAGESf. Additionally, face AEBF in Fig. 3(c) is coincident with the surface of the mesh model, so it is not determined as a member of NAGESf. Furthermore, points of NAGESp and edges of NAGESe solely corresponding with these faces are removed, e.g. line FA and EB in Fig. 3.

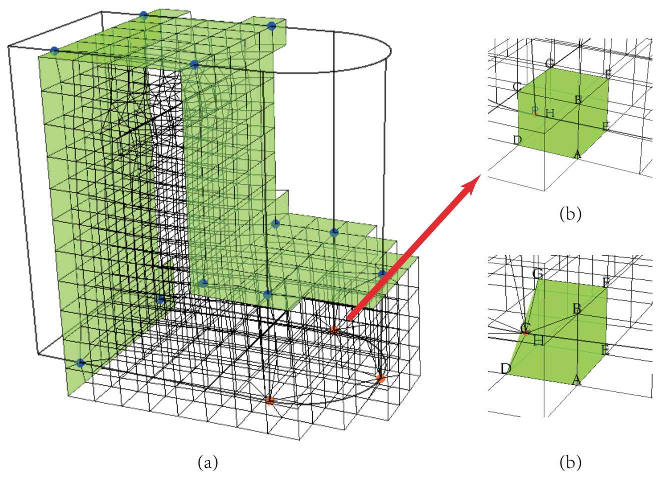

FIGURE 6. Node repositioning for NAGESp: (a)after node repositioning, (b)hexahedron before node repositioning, and (c)hexahedron after node repositioning.

MATCHING OF MESH MODEL TO NAGES

In this section, mesh elements are matched to NAGES through repositioning of their nodes, and this is processed from NAGESp, NAGESe to NAGESf. However, simply repositioning mesh nodes to match NAGES may lead to invalid mesh model or mesh model with bad quality. Therefore, mesh model validation and mesh quality are considered when matching.

Node Repositioning for NAGESp

Mesh quality is considered when selecting mesh nodes to match the points of NAGESp. Here the mesh quality for hexahedron is measured by scaled Jacobian Matrix [8]. If scaled Jacobian Matrix of a hexahedron is less than 0, the hexahedron is distorted; if it is between 0 and 0.2, it is a bad element; hexahedron with scaled Jacobian Matrix between 0.2 and 1.0 is an acceptable element for analysis.

In the process of NAGESp generation, points of NAGESp are computed as the interaction points between geometry entities and mesh elements. Therefore, a point of NAGESp is corresponding to a mesh element (mesh node, mesh edge element, mesh face element or hexahedron) for intersection, and mesh nodes of this mesh element are selected as candidate nodes for repositioning. As shown in Fig. 6(b), there are eight mesh nodes of the hexahedron as candidate nodes for point *P*. Then the worst mesh quality of the adjacent hexahedrons of a candidate mesh node is compared with that of the others after node repositioning, and the best node is selected. According to this rule, node *C* is selected as the best candidate node and moved to the position of point *P* as shown in Fig. 6(c). Finally, mesh nodes are selected and moved to points of NAGESp as shown in Fig. 6(a).

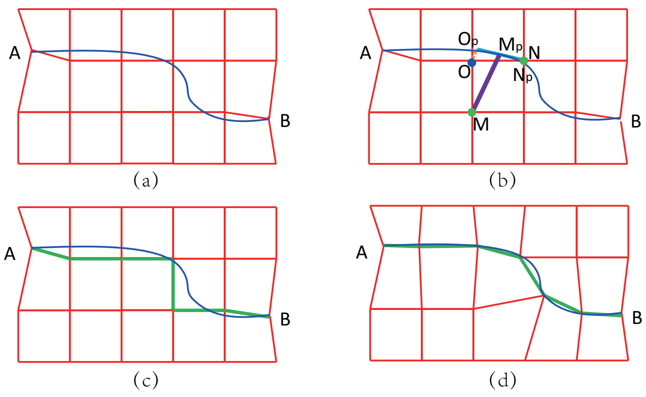

FIGURE 7. Matching mesh edge elements to curve AB on quadrilateral mesh: (a)quadrilateral mesh and curve AB, (b)computation of h(M) and h(N) for node M and N, (c)repositioning path, and (d)mesh edge elements after node repositioning.

Node Repositioning for NAGESe

Node Repositioning Using Shortest Path Algorithm. When matching mesh edge elements to NAGESe, there may be complex interactions between hexahedrons and NAGESe for hexahedral mesh model. Our aim is to match a continuous chain of mesh edge elements to the edge of NAGESe. So shortest path algorithm can be used to identify this chain of mesh edge elements which is denoted as *repositioning path*. In our method, A-Star [9] algorithm is used to identify this repositioning path while ensuring the mesh edge elements close enough to the edge after node repositioning.

NAGESe consists of NAGESge and NAGESil. We focus on node repositioning for NAGESge, because intersection lines of NAGESil can be converted to geometric edges. As nodes are repositioned to the end points of edges of NAGESge, the problem of node repositioning for this type of edge is to find a continuous chain of mesh edge elements, which starts from one end point of the edge to the other end point. In Fig. 7(a), we use a quadrilateral mesh model as an example. There is a curve from node *A* to node *B* on the mesh, and then we move the mesh nodes onto the curve, while the two nodes have already been moved to the two end points of the curve. Additionally, the selected chain of connected mesh edge elements must be close enough to the curve in order to keep the boundary constraints.

Therefore, A-Star algorithm [9] which is used to plot an efficiently traversable path between two nodes in a network, is used to determine the repositioning path. In A-Star algorithm, distance-plus-cost heuristic function is defined for every candidate node *N*:

$$f(N) = g(N) + h(N). \quad (1)$$

In Eqn (1), path-cost function $g(N)$ defines the cost from the starting node A to node N, and distance heuristic estimate function $h(N)$ denotes the estimation cost from node N to target node B. Finally, $g(N)$ and $h(N)$ are defined as follows to ensure the selected path close enough to the curve AB:

$$g(N) = g(O) + \overset{\frown}{O_pN_p}; \tag{2}$$

$$h(N) = O_pN + \lambda NN_p + \overset{\frown}{N_pB}. \tag{3}$$

In the two equations, node O is the previous node of node N. Node O_p is the projection point of node O on curve AB, so does node N_p which is the projection point of node N on curve AB. Besides, $\overset{\frown}{O_pN_p}$ is the distance between node O_p and node N_p along curve AB, and $\overset{\frown}{N_pB}$ is the distance between node N_p and node B along curve AB. In Eqn (1), λ is used to increase the weight of the distance between the candidate node and the curve, so it is set greater than 1.0, and set to 3.0 in our implementation. The computation of distance heuristic estimate function $h(N)$ is illustrated in Fig. 7(b), there are two candidate successor nodes for node O: node N and node M. Then we found that $h(N)$ is less than $h(M)$, so node N is a better node at this state.

For a section of geometric edge starts from the position of node A and ends at the position of node B, node repositioning for curve AB is processed using the following steps according to A-Star algorithm:

1. Set $g(A) = 0$, and compute $f(X)$s of nodes adjacent to node A through a mesh edge element;
2. Select node N with the minimal $f(X)$ from the set of nodes whose $f(X)$ have already been computed;
3. Update $g(X)$s and $f(X)$s of nodes that are previous nodes of node N;
4. Compute $f(X)$s of nodes adjacent to node N through a mesh edge element;
5. Back to step 2. unless node B is adjacent to node N;
6. Move nodes of the best repositioning path onto the projection point on curve AB.

The next problem is to convert intersection lines to geometric edges. This can be done by fitting a chain of intersection lines to a segment of B-Spline, and the intersection points are the interpolation points. However, there may be intersection points with more than two intersection lines of NAGESil or edges of NAGESge connected, e.g. point P and N in Fig. 3(b). So the network of intersection lines are segmented using these intersection points, and they are converted to individual chains of intersection lines and fitted to segments of B-Spline for node repositioning.

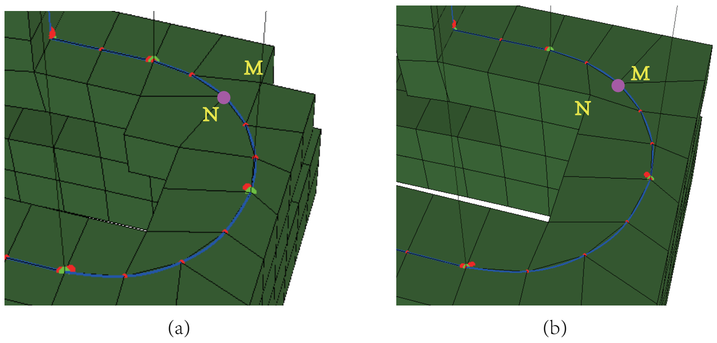

FIGURE 8. Node adjusting for repositioning path: (a)bad mesh quality before node adjusting, and (b)adjust node M to replace node M.

Node Adjusting for Mesh Quality. When using shortest path algorithm for node repositioning, only the mesh nodes with the minimal cost of node moving are selected, but mesh qualities are not considered. Therefore, mesh nodes on the repositioning path have to be adjusted according to mesh qualities. Then, if the minimal value of the scaled Jacobian Matrixes for hexahedrons adjacent to a node is less than 0.2, the minimal values of other candidate nodes are calculated, and the best one among them are selected to replace the original node. As shown in Fig. 8(a), node N is the node selected using shortest path algorithm, and if it is moved onto the intersection line, the scaled Jacobian Matrixes of a hexahedron is -0.165, that means it is a distorted element. In Fig. 8(b), M is moved onto the intersection line instead of node N. The minimal scaled Jacobian Matrix of hexahedrons adjacent to M is 0.593(hexahedrons to be removed are not considered).

Node Repositioning for NAGESf

The boundaries of NAGESf are NAGESp and NAGESe, and they are with mesh node repositioned, so the node repositioning at the internal of NAGESf is discussed here. Our aim is to find a set of mesh face elements that is able to cut off the mesh model with NAGESe as its boundary, and these mesh face elements have to be close enough to the geometric face of NAGESf, then mesh nodes of these mesh face elements are projected to the geometric face. This problem is similar to that of advancing front method used in Paving [10]. In advancing front method, starting from the boundary of the given region which is called the initial front, quadrilaterals are created iteratively and the front is updated, until the front is eliminated which means that the whole given region is covered by the generated quadrilaterals. While in our problem, the mesh edge elements of NAGESe for the given face of NAGESf make up the initial front, and then mesh face elements adjacent to the mesh edge elements of the initial front are found iteratively.

Different from Paving, the mesh face elements are selected

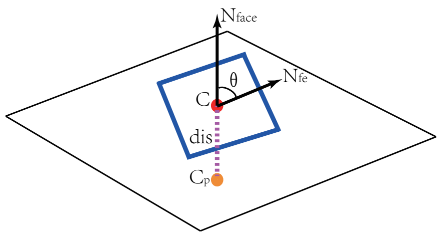

FIGURE 9. Mesh face element evaluation using angle and distance for geometric face.

from existing hexahedral mesh in our method, but not newly created. Therefore, the key issue is to select proper mesh face elements in order to ensure the convergence of the algorithm and the mesh quality after node repositioning. For the convergence of the algorithm, the area of the remaining void region projected on the face must be decreased during the iteration till zero. Then in order to ensure this decreasing, the selected mesh face element after projection onto the face must be at the internal (at least on the boundary) of the current front. Besides, to ensure the mesh quality after projection, we use a metric fe expressed in Eqn. (4) to evaluate the closeness of candidate mesh face elements, and the distance and the normal deviation angle between the mesh face element and the geometric face are considered:

$$fe = \|1 - 2\theta/\pi\| a^{-dis} \tag{4}$$

As shown in Fig. 9, C is the geometric center point of the mesh face element, and C_p is its projection point on the geometric face. θ is the angle between the normal of the geometric face at C_p and the normal of the mesh face element, and dis is the distance between C and C_p. There is a parameter a in Eqn. (4), which is required greater than 1.0, and set to 3.0 in our method. The metric fe is greater when is θ closer to $\pi/2$ or dis is closer to 0, which means the mesh face element is better for node repositioning.

In our method, the mesh face elements adjacent to the front are identified, and those of them whose projections are at the internal (at least on the boundary) of the current front are found as the candidate mesh face elements. Then the mesh face element with the greatest fe is selected from them according to Eqn. (4), and its mesh nodes are moved onto the projection points on the geometric face, at the same time, the front is updated. Finally, the set of mesh face elements are selected until the front is eliminated. In this process, there must be no hexahedron whose neighboring mesh face elements are with the dihedral angle between them greater than 180°, otherwise, for some mesh edges of the front, there may be no candidate mesh face element whose projection is at the internal (at least on the boundary) of the current front. The process of finding mesh face elements is illustrated in Fig. 10 from (a) to (e), and the orange lines are mesh edges of the front. The mesh quality of the mesh face elements after node repositioning are usually low, and Laplacian Smoothing [12] is then performed as shown in Fig. 10(e).

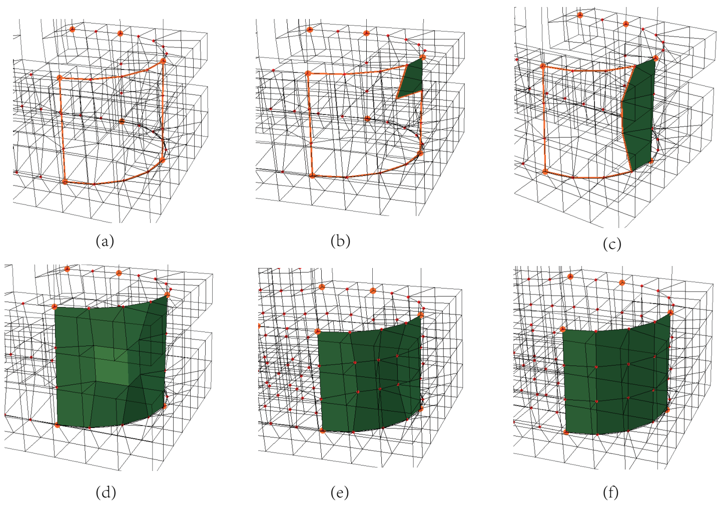

FIGURE 10. Using advancing front method to select mesh face elements: (a)boundary mesh edge elements as initial front, (b)the first selected mesh face element, (c)the first row of selected mesh face elements, (d)selected mesh face elements for this face of NAGESf, (e)move nodes of selected mesh face elements onto the projection points on the geometric face, and (f)smoothing after node repositioning.

HEXAHEDRON REMOVAL AND MESH SMOOTHING

After node repositioning, there is no mesh element intersected with the geometric model. Hexahedrons still inside the geometric models are removed to obtain the mesh model after cutting, as shown in Fig. 1(f).

In the process of mesh node repositioning, mesh elements with bad mesh qualities are inevitable, especially those on the surface of the mesh model are important, as shown in Fig. 11(a). Pillowing [11] is performed to insert an extra layer of hexahedrons from the repositioned nodes. By doing this, the newly added nodes are movable and Laplacian Smoothing [12] is used to smooth the mesh model after hexahedrons removal, and the result is shown in Fig. 11(b).

IMPLEMENTATION AND RESULTS

In the process of node repositioning, *VolumeMesh* developed by ourselves is adopted as the data structure for hexahedral mesh model to find adjacent mesh elements efficiently. Meanwhile, ACIS [13] is used as the geometry engine for the representation of geometric model and associated geometry computing.

In Fig. 12(b), a block-like model is used to cut the mesh

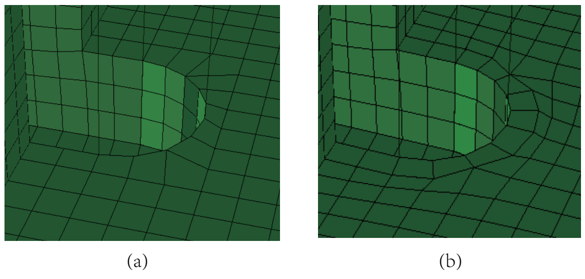

FIGURE 11. Pillowing after mesh elements removal: (a)before pillowing, and (b)after pillowing.

TABLE 1. Scaled Jacobian Matrix after mesh cutting and timing.

	Model in Fig. 1	Block-like model	1/6 nut model
Avg.	0.982	0.990	0.857
Std. Dev.	0.0730	0.0290	0.138
Min.	0.0705	0.530	0.306
Max.	1.000	1.000	0.996
Num.	4857	7571	364
Time(s)	1.124	2.308	0.410

model. In the process, NAGES is generated and can be divided to three parts, and the mesh elements after node repositioning are shown in Fig. 12(c). Because the mesh quality after mesh elements removal is good enough, Pillowing is not performed for this example.

In order to test the robustness of mesh face elements selection for NAGESf, mesh cutting for a nut model is shown in Fig. 13. For convenient, only 1/6 of the model is meshed as shown in Fig. 13(a), and the target model is shown in Fig 13(b) after cutting using a geometric model with the red cone face. In this example, Pillowing is applied to eliminate the doublet hexahedrons as shown in Fig 13(c), and they are smoothed in the final mesh model as in Fig 13(d).

The scaled Jacobian Matrix and time for mesh cutting are listed in Tab. 1 for the example in Fig. 1 and the above two examples, and they are some better than the mesh quality listed in [7]. See from the table, only a small number of hexahedrons with scaled Jacobian Matrix fall below 0.2 for the example in Fig. 1, and there is no distorted hexahedron. Besides, there is no hexahedron with scaled Jacobian Matrix below 0.2 for the block-like model and the 1/6 nut model, and there is also no distorted hexahedron.

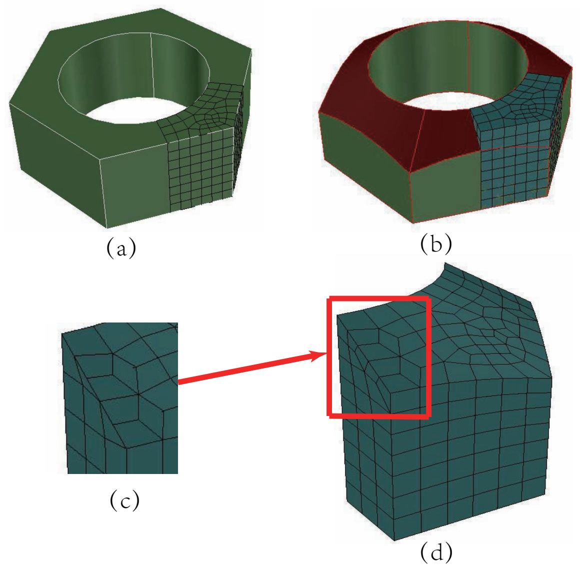

FIGURE 13. Mesh cutting for a 1/6 nut model: (a)a nut model and its 1/6 mesh model, (b)the nut model cut using a cone face, (c)mesh elements with bad qualities before pillowing, and (d)resultant mesh model of 1/6 nut model after cutting.

CONCLUSION AND FUTURE WORK

This paper presented a new approach to hexahedral mesh cutting. The approach allows users to cut a hexahedral mesh model using a complex geometric model to get the required hexahedral mesh model. Compared with the existing methods, the proposed approach has following characteristics:

1. The resultant hexahedral mesh model is more accurate. This is because all the newly added geometric entities resulted from the mesh cutting are generated and well matched by the resultant hexahedral mesh.
2. The geometric model used for cutting as well as the interaction between the hexahedral mesh model and the geometric model are allowed to be more complex. Specifically, the input geometric model is permitted to be partially inside the hexahedral mesh model and its inside part can have arbitrary topological structure.
3. The mesh edge elements that need to be matched to a new added edge are effectively determined using A-Star algorithm.

Regarding the hexahedral mesh cutting algorithm developed in this work, it needs to be further tested by more complex examples and adopt more effective hexahedral mesh optimization algorithm to further improve the resultant mesh quality. In addition, developing of the other hexahedral mesh editing methods such as hexahedral mesh union is also our future work.

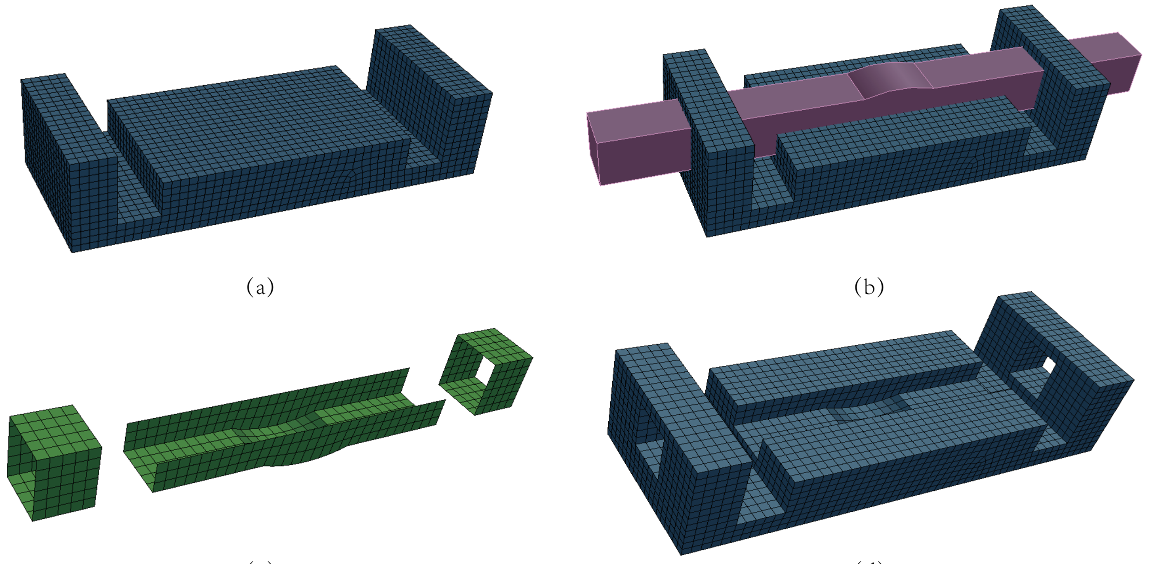

FIGURE 12. Using a block-like model to cut the mesh model: (a)input mesh model, (b)the block-like solid model and the mesh model, (c)mesh elements matched to NAGES, and (d)resultant mesh model.

ACKNOWLEDGMENT

The work described in this paper is supported by the National Science Foundation of China (No. 61173125 and No.60736019).

REFERENCES

[1] Y.S. Liu, H. Zhang, J.H. Yong, P.Q. Yu, and J.G. Sun, 2005. "Mesh blending". *The Visual Computer,21* (11),pp.915-927

[2] Ruding Lou, J.P. Pernot, A. Mikchevitch, and P. Véron, 2010. "Merging enriched Finite Element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance". *Computer-Aided Design,42* (8),pp.670-681

[3] Hua Zhu, Chenghao Hu, and Shuming Gao, 2011. "Finite Element Mesh Editing through CAD Operations". *In Proceddings of 12th International Conference on Computer-Aided Design and Computer Graphics,* pp.53-60

[4] W. Mollemans, F. Schutyser, J. Van Cleynenbreugel, and P. Suetens, 2003. "Tetrahedral mass spring model for fast soft tissue deformation". *Surgery Simulation and Soft Tissue Modeling,* pp.1002-1003

[5] Wenhao Song, and Ligang Liu, 2007. "Stretch-based tetrahedral mesh manipulation". *In Proceedings of Graphics Interface 2007* pp.319-325

[6] Ruding Lou, F. Giannini, B. Falcidieno, J.P. Pernot, P. Véron, A. Mikchevitch, and R. ël Marc, 2010. "Direct Modification of Finite Element Meshes Preserving Group Information". *International Journal of Shape Modeling,16* (1-2),pp.81-108

[7] M.J. Borden, J.F. Shepherd, and S.E. Benzley, 2002. "Mesh cutting: Fitting simple all-hexahedral meshes to complex geometries". *In Proceedings of 8th International society of grid generation conference,*

[8] P.M. Knupp, 2002. "Algebraic mesh quality metrics". *SIAM Journal on Scientific Computing, 23* (1),pp.193-218

[9] Peter E. Hart, Nils J, Nilsson and Bertram Raphael, 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *In Proceddings of IEEE Transactions of Systems Science and Cybernetics,scc4* (2),July,pp.100-107

[10] T.D. Blacker, and M.B. Stephenson, 1991. "Paving: A new approach to automated quadrilateral mesh generation". *International Journal for Numerical Methods in Engineering, 32* (4),pp.811-847.

[11] S.A. Mitchell, and T.J. Tautges, 1995. "Pillowing doublets: refining a mesh to ensure that faces share at most one edge". *In Proceddings of 4th International Meshing Roundtable,* pp.231-240

[12] P. Hansbo, 1995. "Generalized Laplacian smoothing of unstructured grids". *Communications in Numerical Methods in Engineering, 11* pp.455-464.

[13] Jonathan Corney,1997. "3d Modeling using the Acis kernel and toolkit". 1st *ed. John Wiley & Sons, Inc. New York, NY, USA.*