

# Repairing Polygonal Meshes for Volume Meshing

**J.Shen, D.Yoon, Y.Song**

Computer & Information Science Department  
University of Michigan - Dearborn  
Dearborn, MI, USA

**H.Shou**

Dept. of Allied Mathematics  
Zhejiang Univ. of Technology  
HangZhou, P.R.China

**S.Liu**

Generalety LLC  
Livonia, MI, USA

**Abstract** - *In this paper a systematic approach of repairing polygonal meshes is developed for volume meshing, which can then be used in finite element analyses. All poor-quality elements are classified into three general categories: point element, line element, and overlapped element. A robust iterative scheme is proposed to remove all poor-quality elements progressively, leading to a successful volume meshing for finite element analyses. Our method is general, robust, and capable of handling non-manifold surface meshes as long as the input mesh is watertight, which is the basic requirement for volume meshing. Numerical experiments demonstrate the effectiveness of our approach.*

**Keywords:** Mesh repair, Surface mesh, Volume meshing, Finite element analysis.

## 1 Introduction

Transformation from physical objects to digital models is marked as a revolutionary step in human history. Different types of sensors (contact or non-contact) can be used to facilitate such a transformation. After the surface measurement is completed, a surface reconstruction procedure can be initiated to obtain a surface mesh from the measured point clouds. Unfortunately, the conventional surface reconstruction algorithms such as the marching cube algorithm [10] often produced some poor-quality elements in the regions of complex geometry. The resulting surface meshes were then not ready for volume meshing, and therefore could not be directly used for finite element analyses.

Although there are some interactive mesh repairing tools in the market, an interactive repair process could become very tedious if the input mesh model is huge and of complex shape. The volume-based mesh repairing methods [1,6-8,11] in computer graphics sometimes modify the topology of complex structures even at the places where the modification should not occur from a mechanical viewpoint. Note that the topology of a structure is completely different from the topology of the corresponding surface mesh. Therefore, these approaches become meaningless for finite element analyses.

In this paper, we present an automatic scheme for a progressive removal of all poor-quality elements from an input surface mesh with no modification to the topology

and shape of the input structures unless certain region entirely consists of overlapped thin-layer structures, which belong to artifacts generated in a surface reconstruction or simulation process [3,4]. The only restriction to our approach is that the input surface mesh should be watertight, which is the minimum requirement for a successful volume meshing. If the input mesh contains some kinds of cracks due to whatever reasons, readers should first use an existing approach [2,5,9] to fix these cracks.

The main contributions of this paper include:

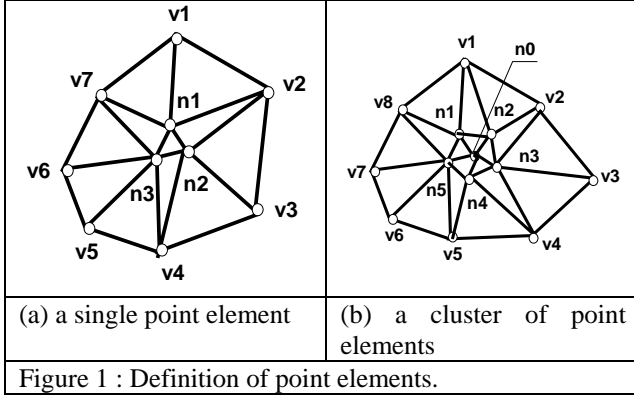
- (1) We classify all the poor-quality elements into three general categories: point, line and overlapped elements. They are viewed as a minimum set that needs to be considered for a successful volume meshing, even though there could be more categories.
- (2) A new automatic scheme that progressively removes all the poor-quality elements defined in our three categories, and leads to a successful volume meshing.
- (3) A special treatment to non-manifold surface meshes.

The rest of this paper is organized as follows. In Section 2, our classification and treatment of poor-quality elements are introduced, and in Section 3 a new iterative scheme for removing all the poor-quality elements in an arbitrary polygonal mesh (manifold or non-manifold) is described. Next in Section 4, a special treatment to non-manifold surface meshes is proposed. Numerical experiments are presented in Section 5, followed by concluding remarks in Section 6.

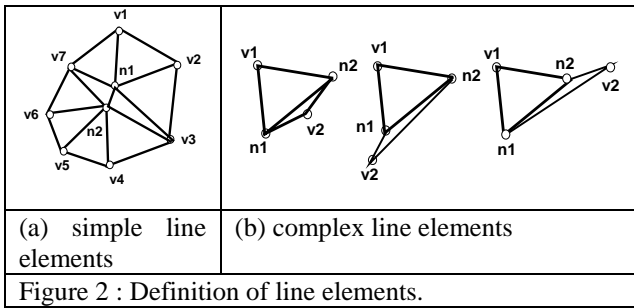
## 2 Classification and Treatment of Poor-quality Elements

It is extremely difficult, if not impossible, to find a unique way to classify all the poor-quality elements of a surface mesh. The solution likely varies and depends upon the context of different applications. For the sake of volume meshing, we proposed to categorize all the poor-quality elements into three groups: point, line and overlapped elements. The three groups form a minimum set that has to be addressed in order to achieve a successful volume meshing.

Point element is defined as an element whose edge length is smaller than a threshold for all its three edges if the element is a triangle, as shown in Figure 1 in which the point elements were scaled up for the sake of clarity. In Figure 1(a), a single point element is formed either by small distances among nodes  $n1$ ,  $n2$  and  $n3$  or by some redundancy of node identification numbers among  $n1$ ,  $n2$  and  $n3$ , i.e.,  $n1=n2$ ,  $n2=n3$ ,  $n1=n3$ , or  $n1=n2=n3$ .



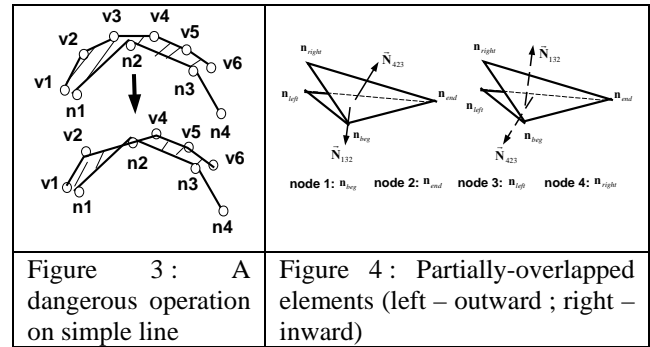
The criterion for identifying a point element is to use a threshold distance, which is equal to  $\lambda_{point} \times l_{avg}$ , where  $\lambda_{point} = 0.1$  and  $l_{avg}$  is an average edge length of the input mesh. For adaptive input meshes,  $\lambda$  should be set to an even smaller ratio. The solution for the point element in Figure 1(a) is to move node  $n3$  to  $n1$  and to move node  $n2$  to  $n1$ . Consequently, four triangles to be deleted include  $\Delta n_1 n_2 n_3$ ,  $\Delta n_1 v_2 n_2$ ,  $\Delta n_2 v_4 n_3$ , and  $\Delta n_1 n_3 v_7$ . In the remaining neighboring elements, replace  $n3$  and  $n2$  by  $n1$ . In the case of a cluster of point elements, as in Figure 1(b), a progressive removal scheme is proposed. For details, refer to Section 3.



Line element is the second category in our mesh repairing system, and consists of two subgroups: simple line and complex line elements, as shown in Figure 2 in which the line elements are scaled up for the purpose of clarity. A simple line element is formed by only one single small-distance edge in an element, as in  $\Delta n_1 v_3 n_2$  of Figure 2(a). The characteristic of a complex line is that three edges

of a triangle element are almost parallel to each other, as illustrated by three variations of  $\Delta n_1 n_2 v_2$  in Figure 2(b).

The criterion for identifying a simple line element is to determine if any edge in a triangle element is smaller than a threshold distance, which is equal to  $\lambda_{line} \times l_{avg}$ , where  $\lambda_{line} = 0.1$ . To remove a simple line element, we move one end node of that small-distance edge to the other end node. In the case of Figure 2(a),  $n2$  is moved to  $n1$ , and triangles  $\Delta n_1 v_7 n_2$  and  $\Delta n_1 v_3 n_2$  are deleted. The removal of simple line elements is a dangerous operation in some special cases such as a thin-wall structure in Figure 3 in which the thickness of a thin wall may become negative due to the deletion of two simple line elements,  $v_2 v_3$  and  $v_3 v_4$ . Note that we demonstrate the elements in Figure 3 on a projected 2-D plane on which each line segment represents a triangle element. For the sake of robustness of our approach, we did not adopt the use of removing simple line elements in our approach.



The criterion for identifying a complex line element is that one of three internal angles of the triangle is greater than  $170^\circ$ . We can consider the complex line element as an extreme case for a sliver element. To remove the complex line element in Figure 2(b), we first calculate the edge lengths of  $n_1 v_2$  and  $n_2 v_2$ . If  $n_1 v_2 < n_2 v_2$ ,  $v_2$  is moved to  $n_1$ ; otherwise,  $v_2$  is moved to  $n_2$ . In addition, the triangle that shares a common edge  $n_1 v_2$  or  $n_2 v_2$  with triangle  $\Delta n_2 v_2 n_1$  should be deleted respectively if  $v_2$  is moved to  $n_1$  or  $n_2$ . For a cluster of complex line elements, a progressive scheme is used as described in Section 3.

Overlapped elements are the third category of poor-quality elements that can be further divided into two subgroups: partially-overlapped and completely-overlapped. Figure 4 is a typical case for partially-overlapped elements. In this figure,  $\vec{N}_{132}$  and  $\vec{N}_{423}$  are the surface normal of triangles  $\Delta 132$  and  $\Delta 423$ , respectively. Our special contribution to the removal of partially-

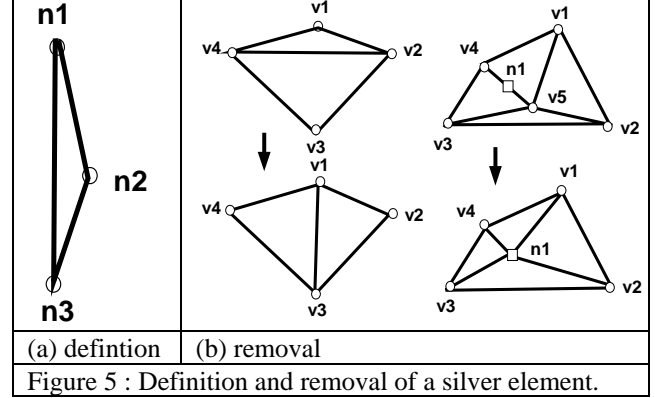
overlapped elements is the distinction between outward and inward elements, as illustrated in Figure 4. To determine if the overlap is inward or outward, assume that there are two adjacent elements  $ie1$  and  $ie2$  whose centroids are  $\vec{c}_{ie1}$  and  $\vec{c}_{ie2}$ , respectively.  $\vec{N}_{ie1}$  and  $\vec{N}_{ie2}$  are the corresponding normal of these two elements. We first calculate  $\vec{V}_2 = \vec{c}_{ie1} - \vec{c}_{ie2}$ . Then, if the dot product,  $\vec{V}_2 \cdot \vec{N}_{ie2}$ , is greater than 0, it is outward; otherwise, it is inward.

The criterion for detecting overlapped elements is that two adjacent elements are inward or outward overlapped elements if the angle between the surface normal of these two elements is greater than  $\alpha_{inward}$  or  $\alpha_{outward}$ , respectively. In this study, we let  $\alpha_{inward} = 150^\circ$  and  $\alpha_{outward} = 165^\circ$ . The rationale for  $\alpha_{inward}$  being smaller than  $\alpha_{outward}$  is that we intend to delete inward overlapped elements in a more aggressive way and to be more tolerant of outward overlapped elements, because the latter is less detrimental to a volume meshing process. If there is a cluster of overlapped elements, a progressive removal is designed, as described in Section 3.

The solution for deleting a pair of partially-overlapped elements, as in Figure 4, is as follows. If edge length  $|\mathbf{n}_{beg} \mathbf{n}_{left}|$  is smaller than edge length  $|\mathbf{n}_{beg} \mathbf{n}_{right}|$ , then move node  $\mathbf{n}_{beg}$  to node  $\mathbf{n}_{left}$ , and move node  $\mathbf{n}_{end}$  to node  $\mathbf{n}_{right}$ ; otherwise, move node  $\mathbf{n}_{beg}$  to node  $\mathbf{n}_{right}$ , and move node  $\mathbf{n}_{end}$  to node  $\mathbf{n}_{left}$ . If node  $\mathbf{n}_i$  is moved to node  $\mathbf{n}_j$ , all the elements that share edge  $\mathbf{n}_i \mathbf{n}_j$  should be deleted, where  $i = beg$  or  $end$  and  $j = left$  or  $right$ .

Besides the above three main categories, a sliver element is another type whose extreme case is a complex line element, as shown in Figure 5. The criterion to determine a sliver element is that one of three internal angles of a triangle is greater than  $\alpha_{sliver}$ , which is  $165^\circ$  in this study. The removal of a sliver element can be pursued by two steps. In step 1, we first test if the surface normals of two adjacent elements are close to each other. This is evaluated by the dot product of these two surface normal vectors against a threshold. If the two normals are close enough and an edge swap will improve the minimum angle of two triangles, we then swap edges  $\vec{v}_2 v_4$  and  $\vec{v}_1 v_3$  in the left part of Figure 5(b). Otherwise, in step 2 we first find which node has the maximum internal angle in the sliver element ( $v_5$  in the right part of Figure 5(b)). We name this node as a base node. Then, a search is conducted to determine the closest neighboring node with respect to this base node. In the case of Figure 5(b), the closest neighbor node that does not belong to the sliver element is  $v_4$ . Calculate the middle point,  $n_1$ , of these two nodes, and

move the base node to the middle point's location, i.e., moving vertex  $v_5$  to  $n_1$  in the right part of Figure 5(b). Note that a precondition for conducting step 1 is the common edge of the two adjacent elements is not shared by a third element. That is, the connection at the common edge is manifold.



### 3 A New Iterative Scheme for Removing All Poor-quality Elements

We propose a new iterative scheme for removing all poor-quality elements progressively, as described in Table 1.

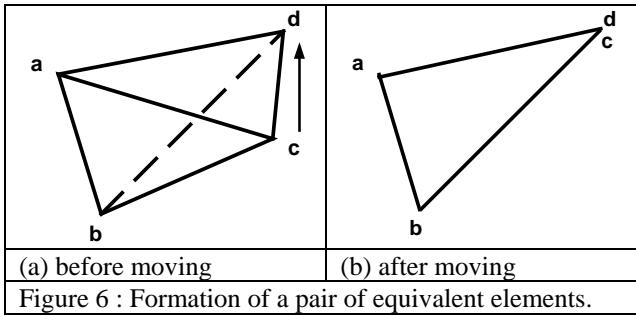
Table I : Algorithm for removing all poor-quality elements

(1) Remove degenerated elements that have at least two same nodes
(2) Remove all the equivalent element sets (definition: Figure 6)
(3) num_remove_elem=1, num_loop = 0
(4) while loop ( num_remove_elem > 0 && num_loop < 100)
{
(4.1) do {
(4.1.1) remove point elements
(4.1.2) num_PE $\leftarrow$ number of removed elements
(4.1.3) if (num_PE > 0)
(4.1.3.1) Remove degenerated elements
(4.1.3.2) Remove all the equivalent element sets
} while (num_PE > 0)
(4.2) do {
(4.2.1) remove complex line elements
(4.2.2.) num_CLE $\leftarrow$ number of removed elements
(4.2.3) if (num_CLE > 0)
(4.2.3.1) Remove degenerated elements
(4.2.3.2) Remove all the equivalent element sets
} while (num_CLE > 0)
(4.3) Remove partially-overlapped elements
(4.3.1) num_POE $\leftarrow$ number of removed

elements (4.3.2) if ( num_POE > 0) (4.3.2.1) Remove degenerated elements (4.3.2.2.) Remove all the equivalent element sets
(4.4) num_remove_elem     number of all the removed elements in current loop
(4.5)    num_loop $\leftarrow$ num_loop + 1 }
(5) Remove sliver elements

Note that we designed a two-layer iteration structure in Algorithm 1 to remove all the poor-quality elements progressively. num\_remove\_elem and num\_loop refer to the number of all the removed elements in the outer loop and the number of the outer loops, respectively. num\_PE, num\_CLE and num\_POE are the numbers of removed point elements, complex line elements and partially-overlapped elements in the current outer loop, respectively.

In this study, a *degenerated element* refers to the element that contains at least two same nodes in that element. In Step 2 of Algorithm 1, an *equivalent element set* means a set in which each element has the same nodes as other elements. Figure 6 is a typical situation, which leads to the formation of a pair of equivalent elements. If you move node **c** to node **d**, two triangles  $\Delta acd$  and  $\Delta dbc$  will be deleted, and other two triangles  $\Delta abc$  and  $\Delta abd$  become a pair of equivalent elements.



Note that there is no inner loop for the removal of partially-overlapped element in Step 4.3 of Algorithm, because this removal frequently generates some line elements that need to be deleted before a next iterate for removing the partially-overlapped elements. In some situations, Step 4.3 does not produce line elements and the outer loop will return back to this step, leading to an elusion of iterative execution of Step 4.3.

We did not let the removal of sliver elements be in the two-layer loop structure. Instead, the removal of sliver elements is conducted only once at the end of Algorithm 1, and the total number of elements is maintained the same in Step 5.

## 4 A Special Treatment to Non-manifold Meshes

A non-manifold surface is in general more difficult to be handled than a corresponding manifold surface, especially when a non-manifold surface is coupled with inward and/or outward partially-overlapped elements, as shown in Figure 7. In this paper, we use a terminology, *non-manifold connection*, to represent a singular point or edge connection where non-manifold occurs. We propose the following principle for handling a non-manifold connection in the context of volume meshing.

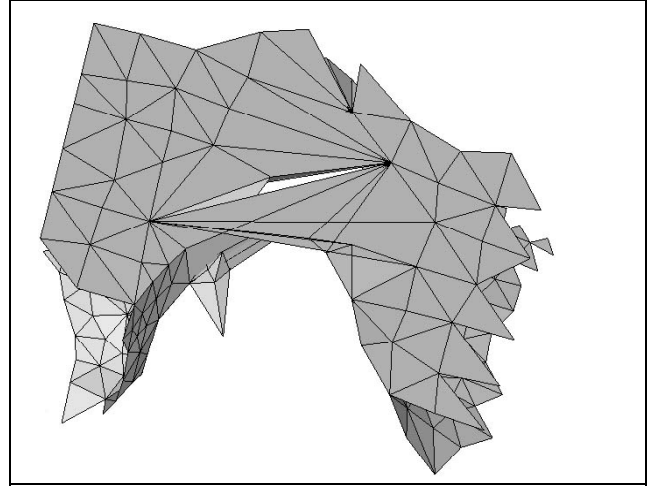
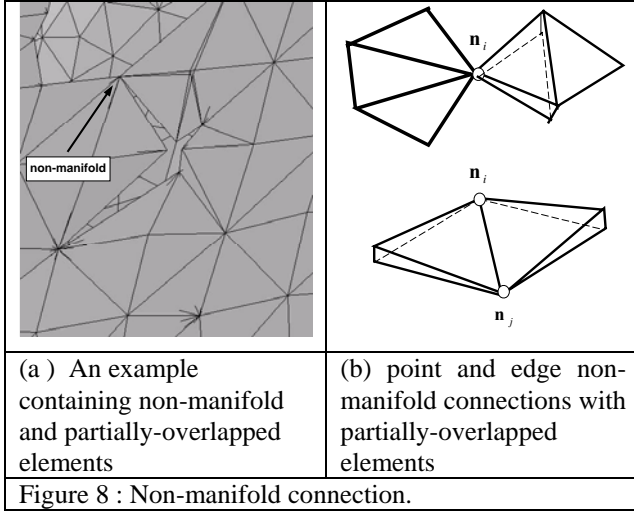


Figure 7 : A failure case of a non-manifold surface coupled with partially-overlapped elements.

### Principle 1: Breakup of Non-manifold Connection

If all the elements that are adjacent to a non-manifold connection are of good quality, we should maintain this connection. If the elements at one or both sides of the non-manifold connection are partially-overlapped, then this connection should be broken apart. □

The implication of this principle is that we should maintain the non-manifold connection where all the neighboring elements are of good quality. This is very important to finite element analyses, because there is no reason to alter the topology of a structure in regular cases. However, there are some special cases in which the topology of a structure needs to be changed. For example, Figure 8(a) shows a case in which we have a thin-layer structure with many outward partially-overlapped elements. The layer is an artifact caused by a surface reconstruction process. It is so thin that these overlapped elements should be removed.



The solution for removing elements at a point non-manifold connection is given in Algorithm 2. Here, the element connectivity means that if two elements share an edge, they are connected. Note that Step 1 in Algorithm 2 is a one-time preprocessing, which can be used for processing the entire surface mesh. Step 4.2 essentially breaks up all the elements, which are adjacent to node  $\mathbf{n}_i$ , into two sets:  $S_1$  and  $S_2$ . Although  $\mathbf{n}_i^*$  has the same coordinates as  $\mathbf{n}_i$ , the elements in these two sets become disconnected after the execution of Step 4.2. It should be noted that a precondition for this disconnection is the existence of partially-overlapped elements to be deleted in either  $S_1$  or  $S_2$ .

The solution for removing partially-overlapped elements at an edge non-manifold connection is slightly more complex, as described in Algorithms 3 and 4. Note that an element may share an edge with several neighboring elements on a non-manifold surface mesh. Step 4.1 in Algorithm 3 is further explained in Algorithm 4. A precondition for Algorithm 4 is that the surface mesh has been properly reoriented.

Table II. Algorithm 2 Removal of Elements at a Point Non-manifold Connection

(1) Calculate element connectivity for all the elements on a surface
(2) Assume a node, $\mathbf{n}_i$ , which is in a partially-overlapped element to be deleted
(2.1) Find a set that contains all the elements adjacent to this node, $Neighbor(\mathbf{n}_i)$
(2.2) Start from an element in $Neighbor(\mathbf{n}_i)$ and perform a breath-first search on the basis of element connectivity.
(2.3) If all the other elements can be reached in the breath-first search, the connection at $\mathbf{n}_i$ is

manifold. Otherwise, it is non-manifold.
(3) If the connection at $\mathbf{n}_i$ is manifold, remove the partially-overlapped element in a regular way described in Section 3
(4) If the connection at $\mathbf{n}_i$ is non-manifold, create a new node $\mathbf{n}_i^*$ that has the same coordinates as $\mathbf{n}_i$
(4.1) Partition $Neighbor(\mathbf{n}_i)$ into two subsets: $S_1$ and $S_2$ . $S_1$ contains all the elements that can be reached from the partially-overlapped element by element connectivity; $S_2 = Neighbor(\mathbf{n}_i) - S_1$ .
(4.2) For each element in $S_1$ , if it contains node $\mathbf{n}_i$ , replace it with $\mathbf{n}_i^*$
(4.3) Remove the partially-overlapped element in a regular way described in Section 3.

Table III. Algorithm 3 Removal of Elements at an Edge Non-manifold Connection

(1) Calculate element connectivity for all the elements on a surface
(2) Assume two edge nodes, $\mathbf{n}_i$ and $\mathbf{n}_j$ , which is on an edge of a partially-overlapped element to be deleted
(2.1) If the partially-overlapped element shares edge $\mathbf{n}_i\mathbf{n}_j$ with only one neighboring element, the connection is manifold.
(2.2) Otherwise, the connection is considered as non-manifold in a loose sense.
(3) If the connection at edge $\mathbf{n}_i\mathbf{n}_j$ is manifold, remove the partially-overlapped element in a regular way described in Section 3
(4) If the connection at edge $\mathbf{n}_i\mathbf{n}_j$ is non-manifold, create two new nodes $\mathbf{n}_i^*$ and $\mathbf{n}_j^*$ , which have the same coordinates as $\mathbf{n}_i$ and $\mathbf{n}_j$ , respectively
(4.1) Partition $Neighbor(\mathbf{n}_i, \mathbf{n}_j)$ into two subsets: $S_1$ and $S_2$ .
(4.2) For each element in $S_1$ , if it contains nodes $\mathbf{n}_i$ and $\mathbf{n}_j$ , replace them with $\mathbf{n}_i^*$ and $\mathbf{n}_j^*$ , respectively.
(4.3) Remove the partially-overlapped element in a regular way described in Section 3.

Table IV. Algorithm 4 Details of Step 4.1 in Algorithm 3.

(1) Calculate a vector $\vec{N}$ along the edge direction $\mathbf{n}_i\mathbf{n}_j$
(2) Make a plane that passes node $\mathbf{n}_i$ with its normal as $\vec{N}$
(3) Project all the elements in $Neighbor(\mathbf{n}_i, \mathbf{n}_j)$ of Algorithm 3 onto this plane, leading to the same number of degenerated edges

(4) Make another plane that passes node $\mathbf{n}_i$ with its normal as the normal of the partially-overlapped element
(5) Partition all the degenerated edges into two subsets: $E_{same}$ and $E_{opp}$ by using this second plane. $E_{same}$ contains all the degenerated edges on the same side as the surface normal of the partially-overlapped element, while $E_{opp}$ contains the rest edges on the opposite side.
(6) Insert the partially-overlapped element into $S_1$ of Algorithm 3
(7) If $E_{opp}$ is not empty, insert the element, represented by the closest edge in $E_{opp}$ from the partially-overlapped element, into $S_1$
(8) Otherwise, insert the element, represented by the farthest edge in $E_{same}$ from the partially-overlapped element, into $S_1$
(9) $S_2 = Neighbor(\mathbf{n}_i, \mathbf{n}_j) - S_1$ .

## 5 Numerical Results and Discussions

The proposed approach was implemented in VC++ and tested on a Sony VGN-S360 laptop computer with an Intel Pentium process 1.7 GHz and 512 MB of RAM. To test the functionality of volume meshing, we chose one of the best commercial tetra-meshers in the market, HyperMesh<sup>®</sup> from Altair Engineering Inc.

With a data model related to Figure 9, volume meshing failed due to self intersection in the input surface mesh. Figure 9 (a) shows a surface mesh after using our approach. The total number of various types of deleted elements is given in the second column of Table V. In this column, we use PE, LE, OE and SE to denote point, line, overlapped and swapped elements, respectively. Figure 9(b) shows the time pattern of these four groups of deleted elements. We use the vertical axis to represent the number of deleted elements, and four lanes to demonstrate a progressive element removal process with the closest lane from readers as PE and the farthest lane as SE. Time step in Figure 9(b) refers to the step in the outer loop of Algorithm 1.

Figure 10 shows a special beam structure. The initial tetra-meshing failed because the boundary face connectivity was wrong. Figure 10(a) shows the surface mesh after mesh repair, which was successfully meshed by HyperMesh into tetra elements, as indicated in the last three columns of Table V. Although the mesh quality in Figure 10(a) is not quite good, the resulted tetra mesh can provide a rough estimation of the mechanical behavior of the structure. If users aim at a high-accuracy finite element

analysis, the surface mesh after the mesh repair should undergo another remeshing process, leading to a surface mesh as in Figure 10(b). How to remesh a surface mesh is beyond the scope of this paper.

Table V : Numerical Experiment Results of Repairing Different Input Meshes

Model	# of deleted elements	Status of volume meshing	# of volume elements	# of volume vertices
Figure 9 # of elems =19846 # of vertices = 9915	874 (PE) <sup>1</sup> 16 (LE) 48 (OE) 2 (SE)	Failure <sup>2</sup> (prior) Success (post)	52672	13518
Figure 10 # of elems =12164 # of vertices =6087	70 (PE) 6 (LE) 158 (OE) 0 (SE)	Failure (prior) Success (post)	31651	8538

<sup>1</sup> PE, LE, OE and SE refer to point, line, overlapped and swapped elements, respectively.

<sup>2</sup> prior and post mean the moments before and after the mesh repairing, respectively.

The importance of the mesh repair algorithm proposed in this paper is two-folded. Firstly, it allows users to conduct a volume meshing and to obtain a quick estimation about the mechanical behavior of the structure; secondly, it removes all poor-quality elements from the input mesh. This facilitates a robust remeshing process, because all the point, line and partially-overlapped elements impose a serious threat to the correctness of intersection calculation and therefore the robustness of the remeshing process.

## 6 Conclusions

In this paper, we propose a systematic classification of poor-quality elements, and a progressive approach for removing all the poor-quality elements with automation and remarkable robustness. Numerical experiments indicate that our approach successfully removed all the poor-quality elements that had an influence on the success of volume meshing. Our approach successfully repaired many complicated surface meshes that failed one of the best commercial meshers in the market, HyperMesh. It provide not only an effective way to handle surface meshes from various sensors or simulation analyses, but also a solid foundation for a robust remeshing process.

## Acknowledgement

This study was supported in part by NSF DMI 0514900.

## 7 References

- [1] Andujar C, Brunet P, Ayala D. Topology-reducing surface simplification using a discrete solid representation. *ACM Transactions on Graphics* 2002; **21**(2):88-105.
- [2] Barequet, G. and Kumar, S. Repairing CAD models. *IEEE Visualization '97*. 1997; 363-370.
- [3] Bendsoe MP. Optimal Shape Design as a Material Distribution Problem. *Structural Optimization* 1989; **1**:193-202.
- [4] Bendsoe MP, Sigmund O. Material Interpolations in Topology Optimization. *Archive of Applied Mechanics* 1999; **69**:635-654.
- [5] Borodin P, Novotni M, Klein R . Progressive gap closing for mesh repairing. In: Vince J and Earnshaw R (eds) *Advances in Modelling, Animation and Rendering*. Springer Verlag:2002; 201-213.
- [6] Dachille, F. and Kaufman, A. Incremental triangle voxelization. *Graphics Interface*. 2000; 205-212.
- [7] Frisken, S. F., Perry, R. N., Rockwood, A. P., and Jones, T. R. Adaptively sampled distance fields: A general representation of shape for computer graphics. *Proceedings of SIGGRAPH*. 2000; 249-254.
- [8] Ju T. Robust repair of polygonal models. *ACM Transactions on Graphics* 2004; **23**(3):888-895.
- [9] Liepa, P. Filling holes in meshes. *Proceedings of Eurographics/SIGGRAPH symposium on Geometry Processing*. 2003; 200-205.
- [10] Lorensen W, Cline H. Marching Cube: A High Resolution 3-D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics Proc*. 1987; **21**(4):163-169.
- [11] Nooruddin FS, Turk G. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics* 2003; **9**(2):191-205.

