Real-time Volumetric Cutting with Mesh Quality Maintaining

Wenhao Yu[a] , Lixu Gu[b,*]
[a]*School of Software, Shanghai Jiaotong University, P.R China*
[b]*Med-X Research Institute, Shanghai Jiaotong University, P.R China*

Abstract
An efficient, accurate and quality maintaining approach of modeling incision in volumetric models is critical in surgical simulation. The proposed incision schema identifies and splits the incision faces and applies an edge-merging algorithm to suppress the creation of redundant tetrahedrons. Then a mesh maintenance scheme composed of a smoothing and a swapping technique is proposed to improve the mesh quality. The two schemes are tested separately in different scenes. Experimental results are shown to demonstrate the effectiveness of our schema for cutting in volumetric models with a low complexity and high quality mesh and its suitability in surgical simulation.
Keywords: Tetrahedral meshes; Real-Time Cutting; Surgical Simulation; Quality Maintenance;

* Corresponding author
Email addresses: stacormed@gmail.com (Wenhao Yu), gulixu@sjtu.edu.cn (Lixu Gu)

1. Introduction and Related Work

Modeling incision on deformable models in real time is of fundamental importance in a wide spectrum of applications, especially in the field of surgery simulation. Due to the specific requirement of virtual surgery, the incision algorithm should be accurate, time-efficient and robust which poses a great technical challenge. To precisely model the incision, elements along the incision trajectory have to be modified which, may create ill-conditioned elements and increase the mesh complexity. At the same time, mesh quality preservative schemes may lose accuracy. Many research works have been conducted to address some of the problems while few of them were proved to satisfy all the conditions.

In the early stage of this research field, incision algorithms were applied to surface meshes. N. Han-Wen and George M. Turkiyyah presented approaches of cutting in triangulated surfaces by subdivide the triangle elements along the scalpel path [1] [2]. And an idea of rearranging the vertexes to fit the cutting path was presented in [3]. It is clear that single surface model is not well enough to realistically represent a deformable model, thus C. D. Bruyns introduced a scheme for accurate surface cutting which includes single surface, multiple surface and hybrid surface [4].

Since many objects used in virtual surgery such as flesh, vessel and organs can be better represented by volumetric models, an incision algorithm for volumetric models is essential. A voxel-based volumetric model can naturally represent the organ structure since it has the same data organization as the 3D digital images produced by medical scanning technologies such as MRI or CT. An algorithm to cutting into this kind of model can be found in [5] and [6]. The connections between neighbor voxels are split up during cutting. But voxel-based representations may bring bumpy intersection faces.

Another type of volumetric models would be to use polyhedron-based volumetric models. Most previous research focused on tetrahedral mesh which is topologically simpler than other volumetric primitives. An instinctive scheme of cutting into tetrahedral mesh model would be to remove all the primitives lying on the cutting path [7][8], which provides real-time simulation yet the volume loss would create jagged surface even though C. Forest proposed a “Refine and Remove” strategy to reduce the artifact[8]. N. Han-Wen and Cakir, O. proposed schemes that moves the vertexes along the cutting trajectory and then subdivides the mesh [9][10], which may create sliver elements. D. Bielser introduced methods that decompose tetra into smaller elements which may create redundant tetrahedrons [11][12], as 17 smaller sub-elements created after the incision in [12]. To cope with this problem, A. Mor used a “Minimal Set Cutting” strategy to improve the situation [13]. However, it still creates ill-conditioned tetrahedral that impose severe time step restrictions. To address the problem brought by sliver elements, algorithms such as edge-collapse [14] and local relaxation [15] are proposed to eliminate degenerate tetrahedra. But these techniques focus on single element information rather than the global mesh quality.

To have a more general solution to eliminate sliver elements, we sought inspiration from research works on global mesh improvement. In [16], the author defines 5 configuration for tetrahedral and then check all the tetrahedra for possible transformation between different configurations. However, it couldn’t eliminate all the bad-shaped tetrahedra. In [17] and [18], a mesh smoothing method called Laplacian Smoothing is proposed which moves the grid point to the geometric center of the adjacent vertexes. Laplacian Smoothing is computational inexpensive yet it doesn’t guarantee improvement in mesh quality. To address this problem, some other works proposed modification to the Laplacian Smoothing algorithm to make sure that mesh quality are improved [19][20][21]. Smoothing techniques

are computational efficient, but their effectiveness in improving mesh quality is restricted by the shape of the model and the number of vertexes. Thus, in [21], a swapping face method was also proposed which reconnects the tetrahedrons separated by a single interior face.

Inspired by the idea presented in [13], a novel incision scheme is proposed in this paper. The concept flow chart is illustrated in Figure 1. After the position of collision detection is detected to provide the incision faces, the incision faces are analyzed and subdivided. We then detect and subdivide singleton vertexes. Finally, we applied an edge-merge algorithm to eliminate elements too small. The scheme could achieve effective performance, but it may also create ill-conditioned elements that damage the stability of simulating system, hence a mesh quality maintenance scheme is designed to improve the quality as well as to preserve the shape of the mesh. Considering the time-efficiency requirement, we combined the Smart Laplacian Smoothing described in [20] and the swapping face method to improve our mesh after a complete incision is performed. We also propose a Boundary Laplacian Smoothing to pre-process the mesh for the other two algorithms. The scheme could achieve significant improvement in mesh quality.

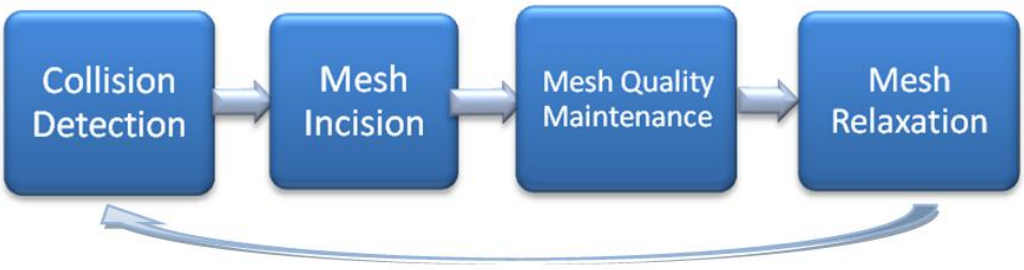


Figure 1: The entire simulation pipeline

2. Volumetric Incision

2.1 Overview

An efficient scheme is proposed for modeling incision in tetrahedral meshes while maintaining a low increment in element number. Incision faces are first identified from the collision detection results and then subdivided progressively from the ones near the surface. After all the incision faces are subdivided, a singleton vertex detection algorithm is applied to find and subdivide the singleton vertexes. With these three steps, the mesh is subdivided according to the tool trajectory, but it may contain a lot of small or bad-shaped elements. To address this situation, we propose an edge-merge algorithm that can effectively eliminate tetrahedrons that are too small or ill-conditioned.

2.2 Identify Incision Face

Each time a collision happens, a collision point would be generated. A tetrahedron can be divided into two smaller tetrahedrons according to the collision point. Thus, a cut through a tetrahedron would create three or four collision points which can then form an incision face as shown in Figure 2.

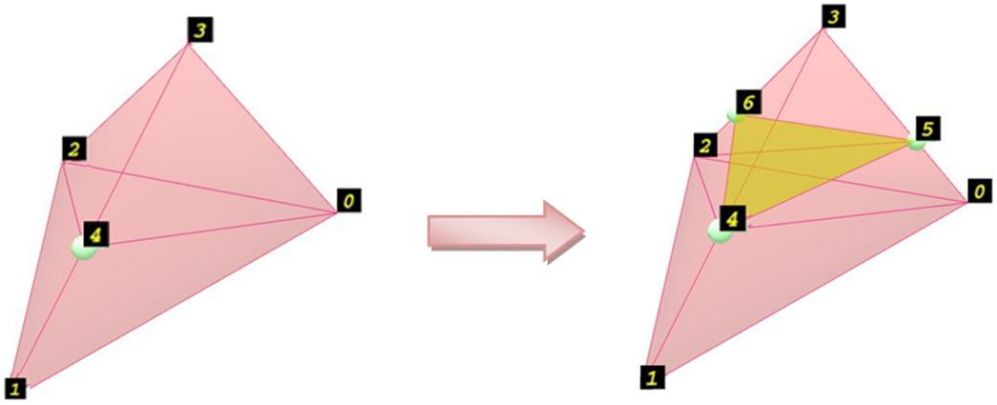
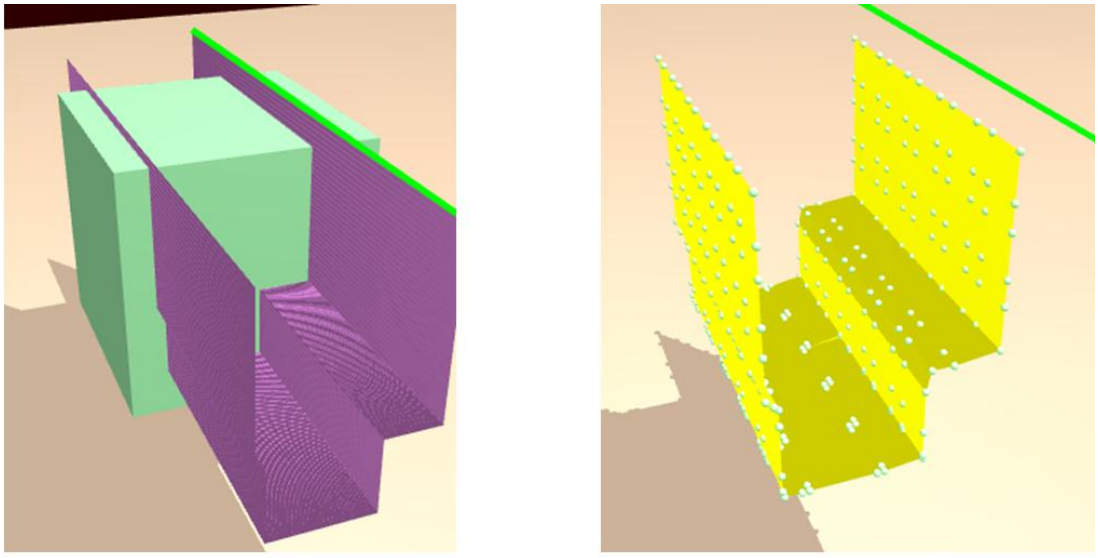


Figure 2: Subdivision of one Tetrahedron (Green ball denotes colliding point and Yellow triangle denotes the incision face).

If we put all the incision faces inside the elements together, we can see a complete collection of incision faces that approximates the tool trajectory (Figure 3). Then these incision faces are subdivided to model the incision of the mesh.



a) tool trajectory b) generated incision faces

Figure 3: Incision faces (yellow surface) approximating the tool trajectory (violet surface). The green spheres are vertexes to be subdivided.

2.3 Mesh Subdivision

After we have gathered the incision faces created by the tool, the mesh subdivision algorithm is applied to subdivide the mesh through the incision faces. We first define the 'dividability' of a vertex. Being dividable means a subset of the cutting faces containing the vertex can either form a circle (type-I) or a sector with two surface edges (type-II) as illustrated in Figure 4 which also shows that the type-I dividable vertex can be converted to type-II dividable vertex by subdividing one of its adjacent vertexes lying on the circle. Since cutting of the model always starts from the surface of the object, type-I dividable vertex will always be converted to type-II after some iterations of division. And we can focus on type-II dividable vertex.

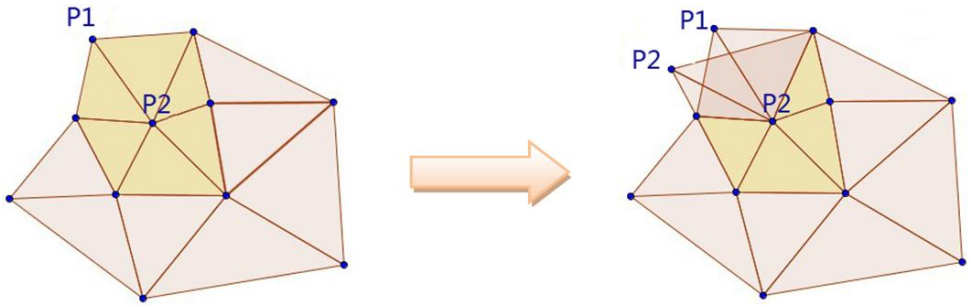


Figure4: left: P2 is type-I dividable because all triangles containing it form a circle. P1 is type-II dividable because the two triangles containing it form a sector with two surface edges.

right: After P1 is divided to P1 and P1', the triangles around P2 are now forming a sector which has two surface edges created by the division of P1.

To find the sector that makes the vertex dividable, we first find the two cutting edges of the sector and check the cutting faces to find all cutting edges. A vertex is a potentially dividable vertex if it is shared by two cutting edges. Then we check all cutting faces containing the potentially dividable vertex to see whether they can form a sector. The pseudocode below describes our implementation:

Algorithm 1: Find All Dividable Vertexes()
1: For each face in cutting face array
2: If it contains a surface edge e
3: Add e to the cutting edge array;
4: For each two edges in cutting edge array
5: If vertex v is shared by both of the edges
6: Add v to the potential dividable vertex array;
7: For each vertex vp in potential dividable vertex array
8: Find all the incision faces containing vp;
9: If a connecting path can be found between each two of the incision faces
10: Add vp to the dividable vertex array;
11: Return dividable vertex array;

When performing the subdivision of a vertex, we first find the circle or sector consists of incision faces that make the vertex dividable, where each face can only be owned by two adjacent tetrahedrons. The tetrahedrons containing the target vertex can be classified into two groups according to which direction they are to the incisions faces. Then, we duplicate the vertex to be subdivided and assign one group of tetrahedrons to it. If a vertex not dividable is subdivided, problem would occur when grouping the tetrahedrons containing the vertex because we don't know whether a tetrahedron should be connected to the old vertex or to the duplicated one.

2.4 Singleton Vertex Subdivision

The subdivision of a vertex would eliminate some of the incision faces, which might cause some vertexes to become singleton vertexes. Singleton vertexes are vertexes that connect two group of tetrahedrons without any faces between, such as the vertex A and B in Figure 5. To address this problem, we proposed a vertex subdivision algorithm.

To identifying the singleton vertexes, we store the number of tetrahedrons connected to each vertex

in a global array. Then we select an arbitrary tetrahedron containing the vertex and count the number of tetrahedrons that can be reached by the initial one through faces containing the target vertex. The two numbers are compared to decide whether a vertex is singleton. Algorithm 2 shows the proposed algorithm. Algorithm 1 and Algorithm 2 is recursively executed until all dividable faces are properly divided.

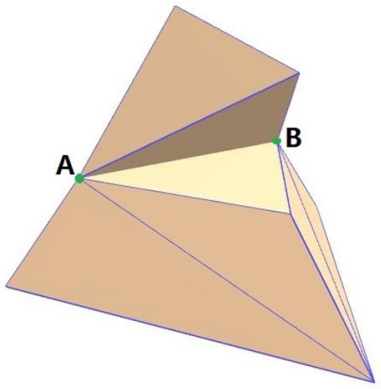


Figure 5: Vertex A and Vertex B should be divided.

Algorithm2: Singleton Vertex Subdivision()
1: Find an arbitrary tetrahedron containing the vertex and add the tetrahedron to a list.
2: While (no new tetrahedrons added to the list)
3: Add all tetrahedrons reachable to the first tetrahedrons to the list.
4: If (size of list < number of tetrahedrons containing the vertex)
5: Create a new vertex and assign it to all tetrahedrons in the list.

2.5 Edge Merging

A result of the incision scheme so far can be seen in Figure 7 (a) where a lot of tetrahedrons with too small size or bad shape can be seen which would increase the burden of the simulation module. Thus, an edge merging algorithm is proposed to eliminate them.

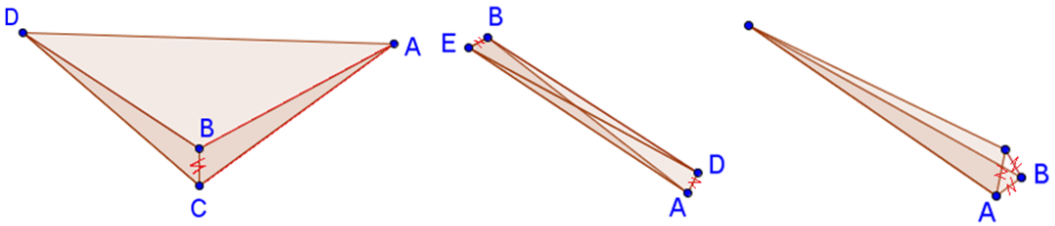
The proposed algorithm is inspired by the 'edge collapse' algorithm presented in [14]. Both algorithms eliminate edges to suppress the mesh complexity. But the edge collapse algorithm only considered elements that are globally too small while elements that are small locally are also considered in our edge merging algorithm. In addition, the proposed algorithm also managed to preserve the shape of the mesh and prevent the creation of singleton vertexes.

An element might be miniature not to the whole mesh, but to its neighbor elements which means the element is negligible but not small in size. There are only three possible cases as shown in Figure 6. Common properties they hold are that they're all bad-shaped and the ratio between the longest edge and shortest edge is very large, thus two thresholds representing the two properties can be used to identify these elements.

To preserve the shape of the mesh, vertexes are divided into three groups: inside vertexes, surface vertexes and contour vertexes. Inside vertexes are moved arbitrarily as long as it doesn't damage the topology of the mesh while surface vertexes are moved along the surface and contour vertexes are moved along the contour. When performing modification, we simply move vertexes to the new position. Though a more accurate strategy could be used to preserve the volume of the mesh as the one described in Section 3.4, it is not needed in this algorithm since the edge we eliminate are usually

negligible.

Our edge merging algorithm also prevents the creation of singleton vertexes. Consider the tetrahedron in Figure 6-a, we assume that the other tetrahedrons containing BC is not going to cause the new vertex to be singleton vertex so we can focus on this one element and to further simplify the problem, we only consider face ABC. Since face BCD is symmetric to ABC, the conclusion from ABC can be easily extended to BCD. We can observe that to create a singleton vertex, edge AB and edge AC must be on surface and whether face ABC, ABD and ACD is on surface should also be considered. The deducted conditions for a safe edge-merging considering face ABC is listed in Figure 6.



a) one short edge b) two short edges c) three short edges

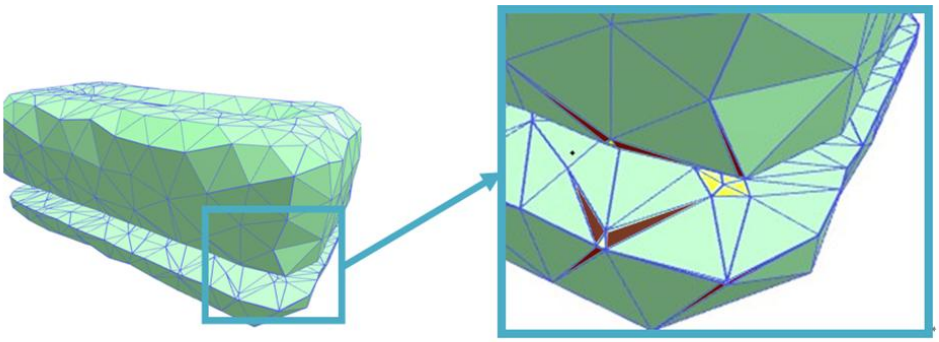
Figure 6: Three shapes that a tetrahedron should be eliminated regardless of the size. Edges marked the red 'z' is to be eliminated.

Considering the creation of singleton vertex, in figure (a) BC can be safely eliminated when:

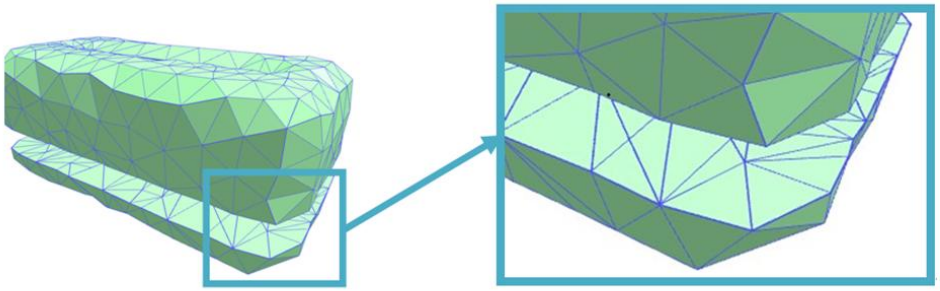
1) (not Surf(BA)) or (not Surf(CA))
2) Surf(BA) and Surf(CA) and Surf(ABC) and ((not Surf(ABD)) or (not Surf(ACD)))

Extension to BCD can be easily done by replacing BA with BD and AC with DC in above conditions.

A comparison between subdivided mesh with and without edge merging is shown in Figure 7. It clearly demonstrates that the edge merging algorithm can effectively eliminate bad-shaped and small-sized tetrahedrons.



a) Mesh Subdivision without edge merging. The red ones are bad-shaped tetras and yellow ones are tetras too small.



b) Mesh Subdivision with edge merging.

Figure 7: Subdivision of tetrahedral mesh with and without edge-merging. As shown in (b), the ill-shaped tetras as well as ones with too small size are successfully eliminated.

3. Mesh Quality Maintaining

In this section, a mesh maintenance scheme is introduced that consists of three algorithms: Smart Laplacian Smoothing, Face Swapping and Boundary Laplacian Smoothing. The first two algorithms adjust the tetrahedrons inside the mesh and the third one modifies the surface of the mesh. The scheme is applied after a complete incision is performed to gain better effect and efficiency.

3.1 Mesh Quality Measurement

Measurements such as dihedral angles, ratio between insphere radius and circumsphere radius, ratio between longest and shortest edge, etc, have been used to represent the quality of an element. In this paper we use the ratio between the longest edge and insphere radius as our measurement. It takes time to compute the insphere radius, but it's efficient to compute the longest edge. So it's a compromise between accurate and efficient. The best case for a tetrahedron would be the regular tetrahedron for which the value of our measurement is $2\sqrt{6} \approx 4.899$. A closer value to this number denotes a better quality.

3.2 Smart Laplacian Smoothing

The Smart Laplacian Smoothing relocates the grid point only if the relocation improves the local mesh quality. The modified vertex position is computed as:

$$P_{new} = \frac{\sum_{i \in V} P_i}{|V|}, \quad (1)$$

where V is the set of incident vertexes of the vertex to be modified, Pi is the position of the ith vertex.

The implementation of Smart Laplacian Smoothing is described in Algorithm 3.

Algorithm 3: Smart Laplacian Smoothing
1: For each divided vertex v
2: Find all tetrahedrons around v
3: Compute the average quality of those tetrahedrons
4: Compute the geometric center of the vertexes connected to v and assign the position to v.
5: Compute the new average quality of the tetrahedrons.
6; Compare the old quality and the new quality. If the new quality is better, use the new position. If not, assign the old position to v.

3.3 Face Swapping

Face Swapping algorithm reconnects a convex consists of 5 vertexes. There're a lot of tetrahedral configurations possible with 5 vertexes, but since we don't want tetras to be self-collided, we can rule out most cases and only consider two types of transformations and three types of configurations as shown in Figure 8. During the face swapping algorithm, tetras forming the two configurations are identified and their qualities are evaluated before and after the transformation, then we keep the configuration with better quality. This algorithm can effectively eliminate some flat tetrahedrons that the smoothing method can't.

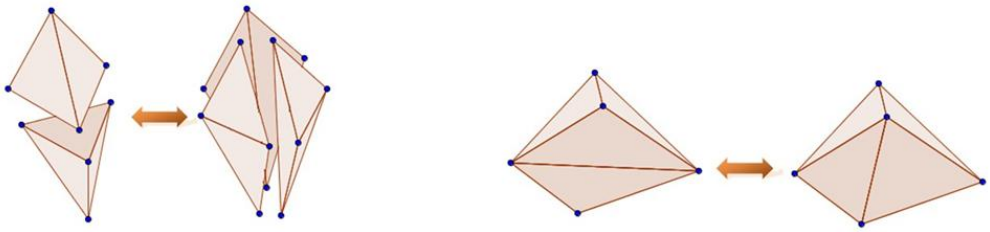


Figure 8: Two possible transformations and three configurations with 5 vertexes.

3.4 Boundary Laplacian Smoothing

No surface vertexes are modified in the previous two algorithms to preserve the shape. But this consideration might impose limitation to the effectiveness of the algorithms.

To solve this problem, we apply a Boundary Laplacian Smoothing to optimize the contour and the incision face before using the other two algorithms.

When optimizing the contour vertexes, the geometric center method wasn't approved because it would shrink the contour. Instead the distance between the vertex to modify and the line constructed by the two neighbor vertexes was kept thus the total area of the contour is maintained. The new position of an edge vertex is computed as:

$$\overrightarrow{nd} = \text{normalize}(<< \overrightarrow{(ori, a)} \times \overrightarrow{(a, b)} >\times \overrightarrow{(a, b)} >) \quad (2)$$

$$dist =< \overrightarrow{(mid, ori)} \cdot \overrightarrow{nd} > \quad (3)$$

$$P_{new} = P_{mid} + dist * \overrightarrow{nd} \quad (4)$$

where P_n is the position of vertex n, $\overrightarrow{(n,m)}$ is an vector pointing from P_n to P_m, P_{mid} is the middle point of the neighbor vertexes. Figure 9 illustrates how vertex C is moved to E instead of D.

Likewise, when optimizing the incision face, a normal Laplacian Smoothing method would shrink the volume of the mesh. Thus a fit plane is calculated using least squares to approximate the vertexes adjacent to the target vertex. Then the target vertex is moved parallel to this plane. This method is not strictly volume-preservative, but we found it visually good enough. A comparison between results with and without Boundary Laplacian Smoothing is shown in Figure 10.

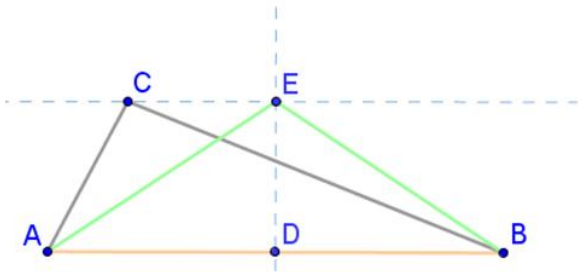
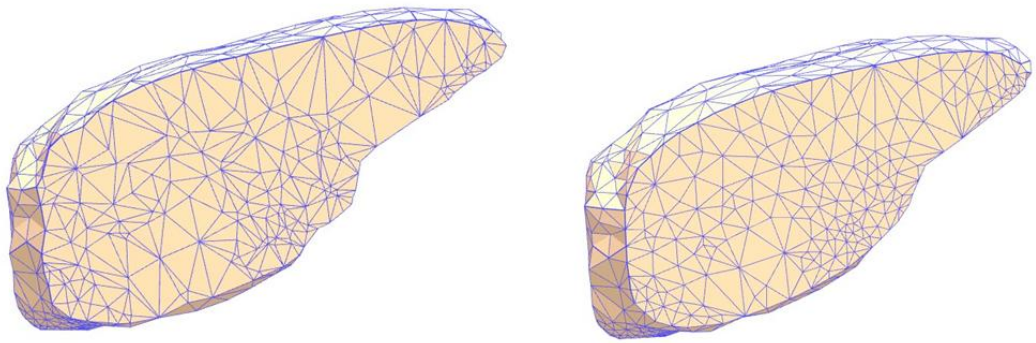


Figure 9: Modified Laplacian Smoothing for contour optimization. Vertex A, C and B are vertexes on the contour. Normal Laplacian Smoothing would use D as the new position for C which causes the contour to shrink. Instead we use E to take place of C.



a) incision surface without BLS b) incision surface with BLS

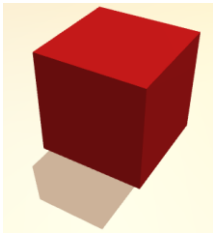
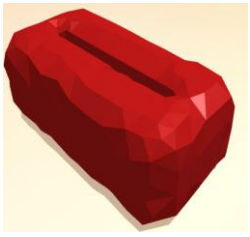
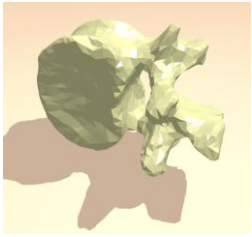
Figure 10: Comparison between incision faces with and without Boundary Laplacian Smoothing

4. Experimental Results

4.1 Test Environments

We test our scheme on a laptop with Intel Core 2, 2.0GHz, 2.0GHz for CPU, 2.0G RAM for memory. We applied our incision and quality maintaining scheme to a set of data with different shape and complexity. The original test data and number of tetrahedrons they contain are illustrated in Table 1.

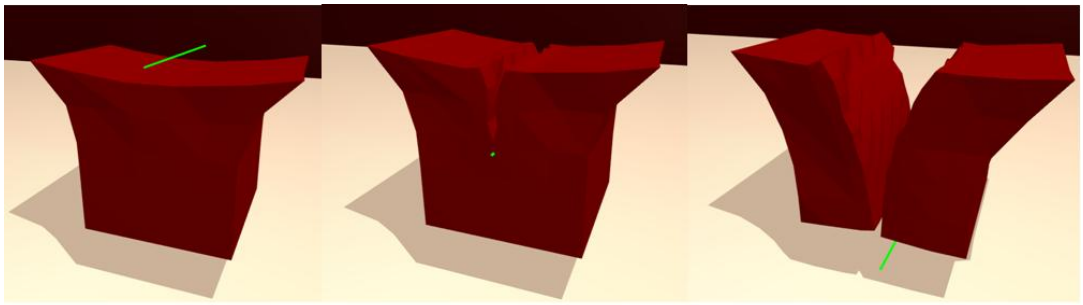
Table 1. The set of data we use to test our scheme.
N: name of the model T: the number of tetra in the model.

	N:CUBE	T:750		N:MEAT	T:1527		N:SPINE	T:7221		N:BUNNY	T:10666

The experiment results for incision scheme and mesh quality maintaining scheme are presented separately. Results for incision scheme would focus on demonstrating its efficiency, scalability and low increment in element number while results for mesh quality maintaining scheme reveal its effectiveness in maintaining the mesh quality.

4.2 Simulation of Incision

The sequence of images presented in Figure 11 shows 3 frames of a cut through Model A where the tetrahedral splits follow the trajectory of the tool. The number of elements after the cutting grows to 1300. Figure 12 depicts a more detailed representation of the subdivided mesh by showing the wire structure of the model. The program runs at a frame rate of around 150fps.



a) b) c)

Figure 11. Three frames of cutting into CUBE.

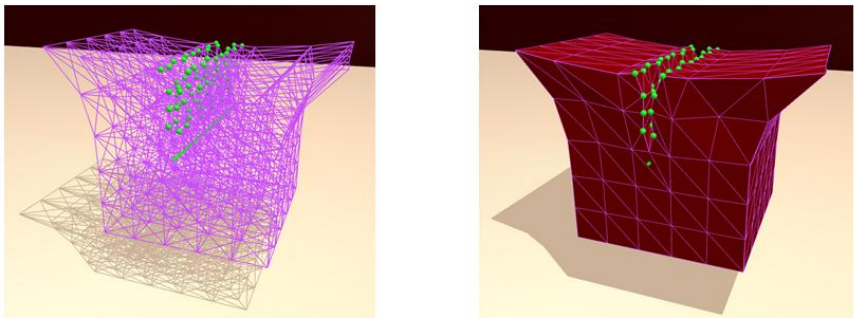


Figure 12: Wire structure corresponding to Figure 11-b. Green balls denote the vertexes subdivided.

To demonstrate the ability of our incision scheme to deal with arbitrary kinds of models and incisions, additional experiments are conducted. As shown in Figure 13, we did a test for a very

complex cutting into model CUBE and other three tests for MEAT, SPINE and BUNNY with an increment in mesh complexity.

Table 2 shows the increment in tetra number, frame per second of the program and performance of the program broken down into different tasks. It reveals that the proposed algorithm is effective, efficient and can notably prevent creation of too many elements. According to the fps data, the program using our volumetric incision scheme can run at 20 fps when dealing with the SPINE model, which is normally sufficient for a virtual surgery application. What's more, the stable percentage of time consumption shows the scalability of our incision scheme.

Table 2. Performance statistics. The Fps is computed over the cut. Notation: RX-Relaxation, CD-Collision Detection, RD-Rendering, GM-Geometry Modifying.

Illustration	Model	Tetra Num Without Edge Merge	Tetra Num With Edge Merge	Fps	Percentage			
					RLX	CD	RD	GM
Fig.8	CUBE	1300	1300	152.3	8.6%	52.8%	30.3%	8.3%
Fig.10-a	CUBE	3840	2418	91.1	8.5%	59.6%	22.5%	9.4%
Fig.10-b	MEAT	2578	2007	66.5	8.1%	34.1%	18.9%	38.9%
Fig.10-c	SPINE	8461	7832	20.2	10.8%	36.8%	9.0%	43.4%
Fig.10-d	BUNNY	12193	11355	13.1	9.6%	40.1%	7.3%	43.0%

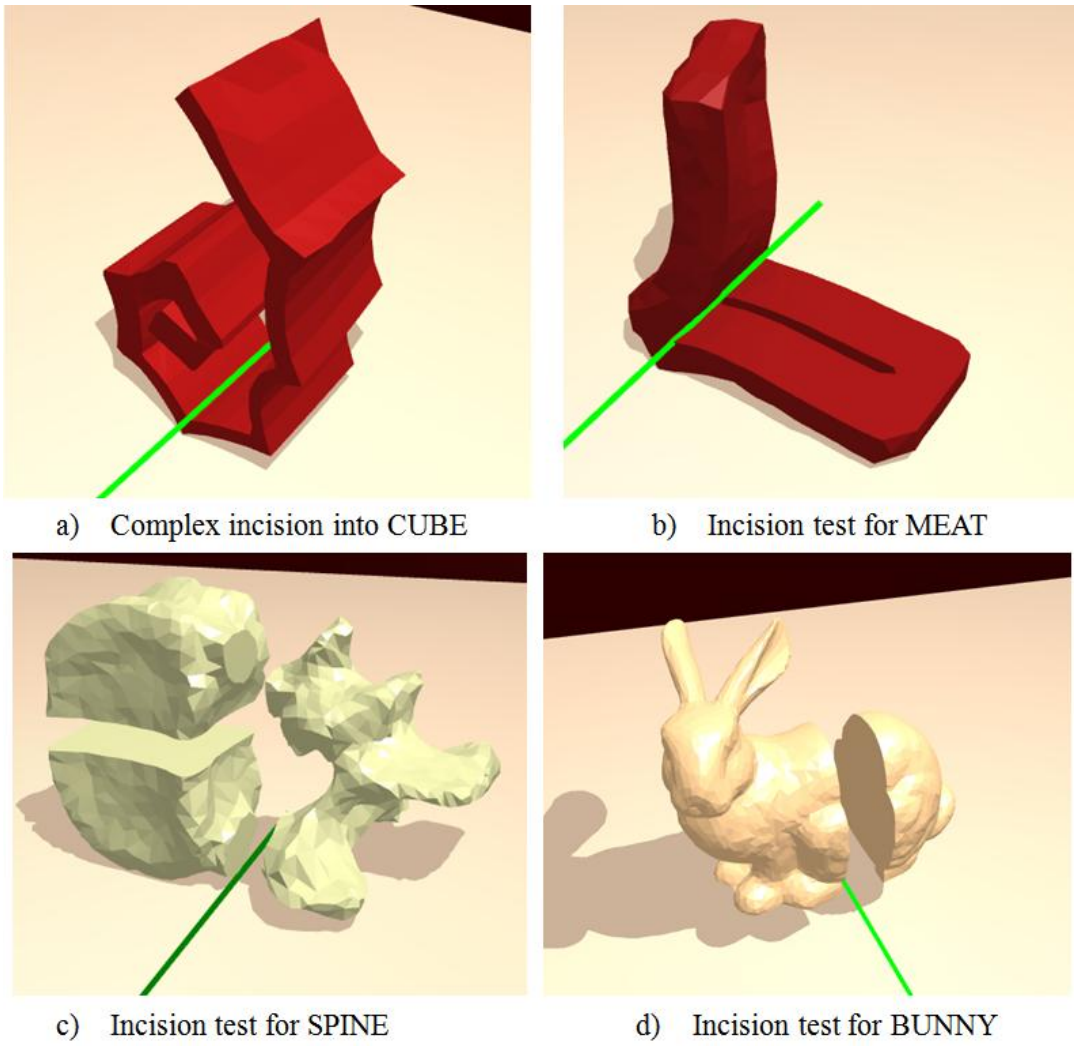


Figure 13 . Additional test cases for incision scheme.

4.3 Mesh Quality Refinement

The test case we use is shown in Figure 14 where the mesh on the incision surface is mostly regular and contains little flat triangles. A more detailed analysis of optimization results is shown in Table 2 containing the maximum, minimum, average value of the mesh quality and the number of bad-shaped elements. The measurement we use here is coherent with the one described in Section 3.1. Since the maintenance scheme is to make the mesh quality close to the uncut mesh, we also evaluated quality of the original mesh. As is shown in Table 3, the average quality of the mesh can be significantly reduced with our maintenance scheme. The quality of the worst element using our scheme wasn't much better, but the number of tetrahedrons with measurement over 200 is much fewer. What's more, the best element in the optimized mesh is even better than the original mesh. To make a more complete illustration of the optimization results, we display the distribution of the mesh quality in Figure 15. As we can see, the percentage of elements with good quality in the mesh with our maintenance scheme is considerably closer to the original mesh than the non-optimized one.

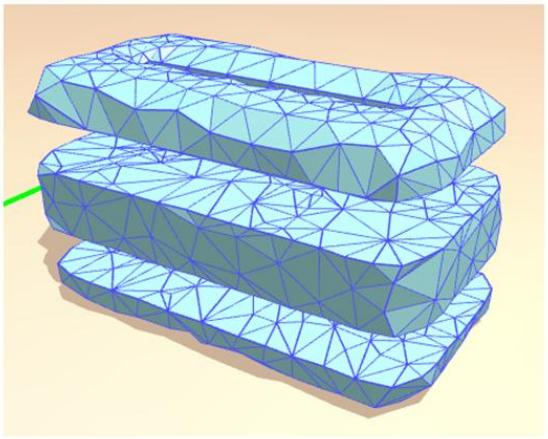


Figure 14: Test case for optimization algorithm.

Table 3. Mesh quality maintenance scheme results.

Mesh	Min	Max	Avg	Percentage of bad-shaped tetra (measurement>200)
Original	7.90	28.25	12.277	0%
No optimization	7.90	850.14	38.27	2.6%
Our Method	7.63	762.98	17.79	0.22%

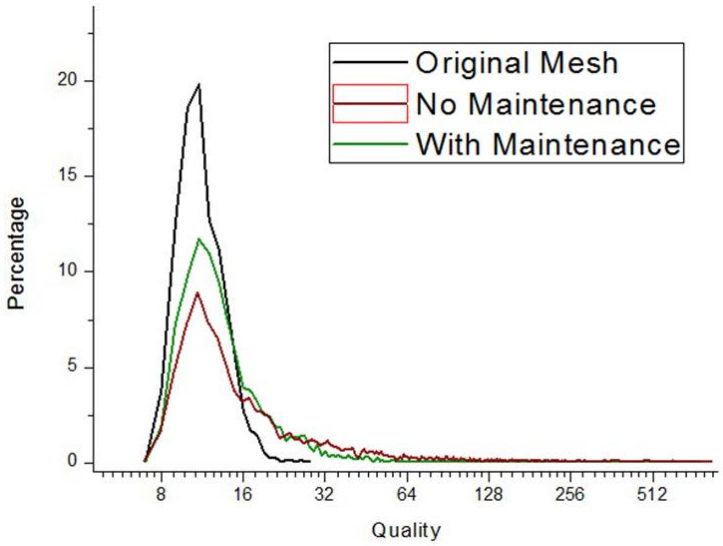


Figure15: Distribution of mesh quality values in the three test cases.

4.4 Application

The proposed schema has been integrated into a practical demo for surgical cutting. The application provides surgeon candidates training in surgical cutting. As is shown in Figure 16, a cut is performed in the middle of a vein model and is correctly modeled by our scheme, where the mesh is efficiently subdivided along the trajectory of the tool and our quality maintaining scheme ensures the stability of the simulation.

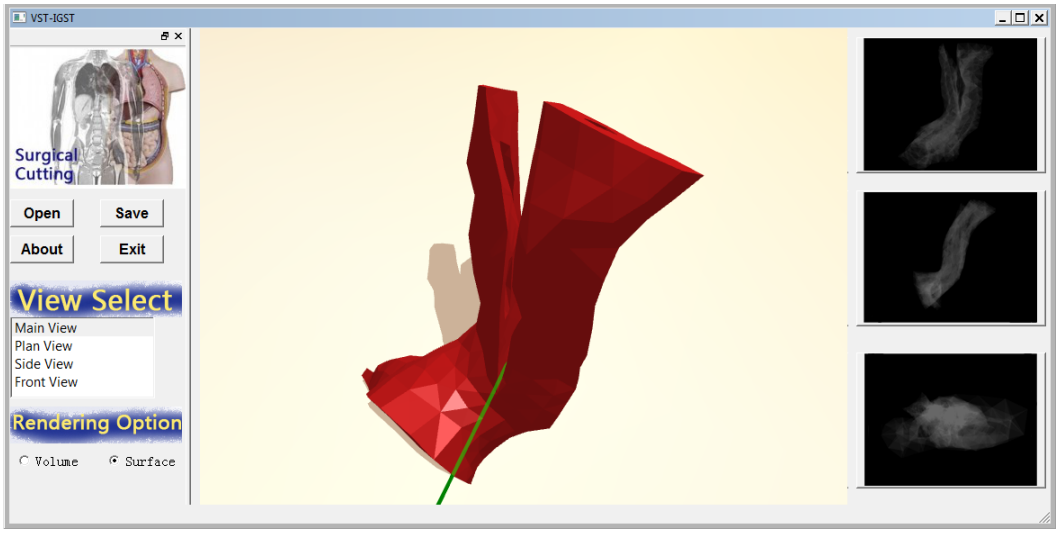


Figure 16. Screenshot of a virtual surgical application which applies our volumetric cutting scheme.

4.5 Discussion

Experiment results have been presented to show the accuracy, time-efficiency and robustness of our incision scheme. Unlike the scheme proposed by D. Bielser [11] which mostly focuses on the accuracy aspect of the simulation, our scheme consider more about mesh quality and complexity, therefore can achieve a better simulation result. When compared to schemes that preserve the mesh complexity by moving vertexes along the incision trajectory such as the one presented in [10], our incision scheme demonstrates more flexibility in dealing with complicated incision as shown in Figure 13-a. Thus our incision scheme is more suitable than these proposed methods in virtual surgery systems where a high mesh quality is demanded and the ability to conduct multiple incisions is required.

Though applicable and practical in many aspects the proposed scheme is, a number of limitations still exist. The proposed incision scheme cannot deal with partially intersected tetrahedrons in that the collision detection is applied to edges which might cause some visual defect when the mesh resolution is low. In addition, the incision faces are split recursively from the contour of the incision, thus the efficiency of the scheme might be lowered if the sweep face created by the tool is very large in one frame. Finally, the proposed mesh maintenance scheme doesn't insert or delete any vertexes, thus the effectiveness of it depends on the topology of the existing vertexes which may prevent the subdivided mesh to achieve perfect quality as the original mesh.

5. Conclusion

An effective scheme is proposed for modeling incision into volumetric meshes as well as the mesh maintenance scheme to maintain the mesh quality. The incision scheme models incision into tetrahedral meshes by subdividing elements along the tool trajectory and an edge-merging algorithm is applied after the subdivision that effectively eliminates redundant elements and keeps a low mesh complexity.

The proposed mesh quality maintenance scheme effectively improves the quality of the subdivided mesh by applying three separate algorithms to the elements near the incision surface. Tetrahedral mesh with better quality ensures a more stable physical system and accuracy of the cutting algorithm.

Results were also presented to show that the scheme can be easily implemented and applied in a virtual surgery system.

As we discussed in Section 4.5, there're still drawbacks in our scheme as the non-progressive incision, recursive splitting and the limitation in mesh maintenance. In addition, there's no force feed-back when performing a cutting. Thus future works will be conducted to address these problems.

Acknowledgements

This research is partially supported by the NSFC research foundation of 61190120, 61190124 and 61271318.

References

[1]. N. Han-Wen and A.-F. van der Stappen. A Delaunay approach to interactive cutting in triangulated surfaces. Algorithmic Foundations of Robotics V, pages 113-130, 2003.

[2]. George M. Turkiyyah, Wajih Bou Karam, Zeina Ajami and Ahmad Nasri. Mesh cutting during real-time physical simulation. Computer-Aided Design, volume 43, issue 7, page 809-819, 2011.

[3]. D. Serby, M. Harders and G. Szekely. A new approach to cutting into finite element models. In Procs. Of the Fourth International Conference on Medical image, page 425-433, 2001.

[4]. C. D. Bruyns and K. Montgomery. Generalized interactions using virtual tools within the spring framework: Probing, piercing, cauterizing, and ablating. In Medicine Meets Virtual Reality 02/10, page 79-85, 2002.

[5]. S. Gibson, J. Samosky, A. Mor, C. Fyock, E. Grimson, T. Kanade, R. Kikinis, H. Lauer, N. McKenzie, S. Nakajima, H. Ohkami, R. Osborne, and A. Sawada. Simulating Arthroscopic Knee Surgery using Volumetric Object Representations, Real-Time Volume Rendering and Haptic Feedback. In Proceedings of the First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery, page 369-378, 1997.

[6]. Pflesser, B., Petersik, A., Tiede, U., Höhne, K. H. and Leuwer, R. Volume cutting for virtual petrous bone surgery. Comput. Aided Surg., volume 7, page 74–83, 2002.

[7]. S. Cotin, H. Delingette, and N. Ayache. ìA hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. The Visual Computer, 16(8), page 437-452, 2000.

[8]. C. Forest, H. Delingette , N. Ayache. Removing tetrahedra from a manifold mesh. In Proceedings of the Computer Animation, page 225–229, 2002.

[9]. N. Han-Wen and A.-F. van der Stappen. Combining finite element deformation with cutting for surgery simulations. In Eurographics 2000, Short Presentations Programme, 2000.

[10]. Cakir, O., Yazici, R. Real-time cutting simulation based on stiffness-warped FEM, Computer and Information Sciences. ISCIS 2009. 24th International Symposium on, page 721 - 724, 2009.

[11]. D. Bielser, P. Glardon, M. Teschner, and M. H. Gross. A State Machine for Real-Time Cutting of Tetrahedral Meshes. In Proceedings for the 11th Pacific Conference on Computer Graphics and Applications, page 337-387, 2003.

[12]. D. Bielser, V. A. Maiwald, M. H. Gross. Interactive Cuts through 3-Dimensional Soft Tissue. Proceedings of the Eurographics '99 (Milano, Italy, September 7-11, 1999), COMPUTER GRAPHICS Forum, Vol. 18, No. 3, C31-C38, 1999

[13]. A. Mor, T. Kanade. Modifying soft tissue models: progressive cutting with minimal new element

creation. In MICCAI, page 598–607, 2000.

[14]. F. Ganovelli, C. O'Sullivan. Animating cuts with on-the-fly re-meshing. In Eurographics 2001, Short Presentations Programme, 2001.

[15]. D. Steinemann, M.Harders, M. Gross, G. Szekely. Hybrid cutting of deformable solids. In Proc. of the IEEE Virtual Reality Conference, page 35–42, 2006.

[16]. Joe, B. Construction of three-dimensional improved-quality triangula-tions using local transformations. SIAM Journal on Scientific Computing 16 (6), page 1292-1307, 1995.

[17]. David A. Field. Laplacian smoothing and Delaunay triangulations. Communications in Applied Numerical Methods Volume 4, Issue 6, page 709–712,November/December 1988

[18]. Lo SH. A new mesh generation scheme for arbitrary planar domains. International Journal for Numerical Methods in Engineering, 21, page 1403–1426, 1985.

[19]. Staten ML, Canann SA, Tristano JR. An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. Proceedings of the Seventh International Meshing Roundtable, Sandia National Laboratories, page 479– 494, 1998.

[20]. Freitag, L., Jones, M.and Plassmann, P. An efficient parallel algorithm for mesh smoothing. Proc.4th Intl.Meshing Roundtable, Sandia National Laboratories, page 47-58, 1995.

[21]. Lori A. Freitag, Carl Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. International Journal for Numerical Methods in Engineering Volume 40, Issue 21, page 3979–4002, 15 November 1997