Removing Tetrahedra from manifold tetrahedralisation : application to real-time surgical simulation

C. Forest *, H. Delingette, N. Ayache

Epidaure Laboratory, INRIA, 2004 route des lucioles, BP 93, 06902 Sophia Antipolis, France

1 Introduction

1.1 Simulation of cutting

In a general-purpose surgery simulator, it is important to model the action of different types of surgical instruments. The complexity of the simulation not only depends on the nature of this action (suturing, grasping, cutting,...) but also on the nature of the anatomical structures on which they apply. Typically, among the different structures, one can distinguished between vessel models, surfacic structures (membranes) and volumetric structures (parenchymatous organs).

This paper is related to the simulation of cutting instruments (electric lancet, scalpel, cavitron) on volumetric tissues. Cutting tissue is one of the most challenging action to simulate, at least for two reasons. First, it involves the modification of topology of meshes, which in itself may be a complex task. Second, it makes most soft tissue deformation algorithms not compatible with real-time requirements.

In general, the implementation of cutting in a surgery simulator first consists in defining a surface of cut created by the motion of a cutting instrument. Then, the cut is simulated by creating a "trench" at the intersection between the surface of cut and the tridimensional structures. Algorithms for simulating the cut of surfaces have been proposed by many authors[BSM+02,VHM+99,SHS01,NvdS02]. The simulation of cutting volumetric meshes, in particular tetrahedral meshes,

* Corresponding author: Clement.Forest@ircad.u-strasbg.fr

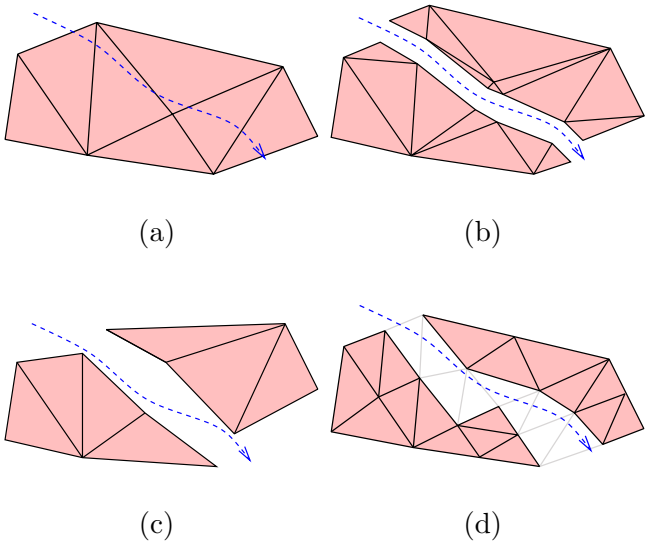


Fig. 1. Different strategies for cutting a triangulated mesh. The cut path is drawn with a dash line. (a) The original mesh; (b) Subdivision strategy; (c) Vertex projection strategy; (d) "Refine and Remove" strategy.

leads to more complex algorithms [BG00,MK00] due to a greater combinatorial complexity.

In figure 1, we have pictured three different strategies for cutting a triangulated mesh (the same strategies apply for a tetrahedral mesh). A common strategy (Figure 1 (b)) consists in subdividing cells (here triangles) along the *cut line or surface* [BG00,MK00,VHM+99,NvdS01]. The drawback of this approach is that it tends to create a large number of little and badly-shaped triangles or tetrahedra which increases the time of computation and may cause instabilities. Recently Nienhuys [NvdS02] has proposed another strategy (Figure 1 (c)) where vertices are projected along the cut surface thus limiting the creation of new elements. However, the algorithm only applies for surface meshes and not for volumetric meshes.

We propose a third strategy consisting of the refinement followed by a removal of the elements located near the surface cut (see Figure 1 (d)). This approach has the advantage of generating high-quality elements but with the drawback of creating cuts that are not very smooth.

In the context of an hepatectomy simulator developped in the Epidaure Project at Inria Sophia-Antipolis, we found this third approach to be acceptable. Indeed, in this case, the cut surface is in general rather irregular. Furthermore, it simulates quite well the action of an ultrasonic lancet (ie. cavitron) which fragilizes and destroys the hepatic parenchyma. The implementation of the third cutting strategy for a tetrahedral mesh requires two basic algorithms : a local refinement algorithm and a tetrahedron removal algorithm. The former

algorithm is based on edge splitting [FDA02] although there exists alternative algorithms[PBKD02].

The latter algorithm describing the removal of tetrahedra from a tetrahedralisation is the focus of this paper. Although this algorithm may seem trivial at first, it turns out to be rather complex when adding the constraint that the mesh must be a manifold.

1.2 Motivations for keeping a manifold mesh

A manifold tetrahedralisation mesh is formally defined in section 2. In essence, a volumetric mesh is a manifold if it has everywhere a thickness strictly greater than zero. When removing a single surface tetrahedron, namely a tetrahedron that has at least one face on the mesh surface, from a tetrahedral mesh, it often creates topological singularities where the mesh has zero thickness. This paper describes an algorithm for removing a single surface tetrahedron while preserving the manifold property of the mesh.

In fact the manifold property is required for performing many low level tasks on tetrahedralisation such as :

- **Computing vertex normal on the mesh surface.** This is especially useful for the visual rendering of mesh surfaces with Gouraud shading or PN-Triangles [VPBM01] but also for haptic rendering;
- **Mesh smoothing.** Many algorithms such as Laplacian smoothing [PS03] are based on the relative position of a vertex with respect to its neighbors. If the link of a vertex is composed of several connected components then Laplacian smoothing may easily create self-intersections;
- **Mesh subdivision.** Subdivision surface and volume [Qin00] algorithms only apply on manifold meshes;
- **Finding all adjacent vertices to a given vertex.** If there are no constraints on the topological nature of the link of a vertex, then one has to use redundant data structure to find vertices adjacent to a given vertex.

Note that the finite element method does not require a tetrahedral mesh to be a manifold. However, non manifold meshes often lead to ill-conditioned linear systems of equations. Furthermore, as experimented in previous implementations, if the manifold property is not enforced on a finite element tetrahedralisation, it is common to observe instabilities around topological singularities.

Finally and most importantly, it turns out that manifold tetrahedralisations, where the link of each vertex is simply connected, are well-suited for determining vertex adjacencies with a minimum amount of information. We have taken

advantage of this fact to propose an efficient tetrahedralisation data structure.

2 Manifold Tetrahedralisation

2.1 Topological description of tetrahedralisations

We first introduce, in a strict topological way, the notions of *tetrahedralisation*.

Definition 1 (Vertex) *Vertices are the basic elements of a tetrahedralisation.*

Definition 2 (Tetrahedron) *A* tetrahedron *is defined by an oriented 4-tuple of four different vertices. Two tetrahedra* $T_1(v_a, v_b, v_c, v_d)$ *and* $T_2(v_e, v_f, v_g, v_h)$ *are identical if there is an even permutation that transforms* (v_a, v_b, v_c, v_d) *to* (v_e, v_f, v_g, v_h)*. If two tetrahedra share the same vertices and if there is an odd permutation between one and another, then they are said to have a different* orientation. [1]

Definition 3 (Edge and Triangle) *If* $T(v_0, v_1, v_2, v_3)$ *is a tetrahedron, the sets* $\{v_i, v_j\}$ *and* $\{v_i, v_j, v_k\}$ *are called respectively* edges *and* triangles *of the tetrahedron.*

Definition 4 (Oriented Triangle) *If* $\{v_1, v_2, v_3\}$ *is a triangle then the oriented triplet* (v_1,v_2,v_3) *is called* oriented triangle*. The oriented triangle* (v_1,v_2,v_3) *is said to be* even oriented *according to the tetrahedron* $T(v_0,v_1,v_2,v_3)$ *and* odd oriented *according to the tetrahedron* $T(v_0,\ v_1,\ v_3,\ v_2)$*.*

Definition 5 (Adjacency) Adjacency *is a* reflexive *property which is set between a tetrahedron, a triangle or an edge, and one of its subset (triangles, edges or vertices). This definition can be extended in the case of two tetrahedra sharing a common triangle.*

Definition 6 (Tetrahedralisation) *A* topological tetrahedralisation *is composed of a set* $\mathcal{V}$ *of vertices and a set* $\mathcal{T}$ *of tetrahedra with the restrictions that a given oriented triangle has* at most *two adjacent tetrahedra, one odd oriented and one even oriented.*

Furthermore, we define the notions of *surface*, *neighborhood*, *connectivity* and

[1] Therefore, an oriented tetrahedron (v_a, v_b, v_c, v_d) has 12 equivalent representation: (v_a, v_b, v_c, v_d), (v_a, v_c, v_d, v_b), (v_a, v_d, v_b, v_c), (v_b, v_c, v_a, v_d), (v_b, v_a, v_d, v_c), (v_b, v_d, v_c, v_a), (v_c, v_a, v_b, v_d), (v_c, v_b, v_d, v_a), (v_c, v_d, v_a, v_b), (v_d, v_c, v_b, v_a), (v_d, v_b, v_a, v_c) and (v_d, v_a, v_c, v_b).

link. These notions are required for the definition of the manifold property.

Definition 7 (Neighborhood) *The* neighborhood of a vertex *is the set of all tetrahedra adjacent to that vertex. The* neighborhood of an edge *is the set of all tetrahedra adjacent to that edge.*

Definition 8 (Surface of a tetrahedralisation) *We define the* surface of a tetrahedralisation *as the set of all triangles adjacent to* exactly *one tetrahedron. A vertex or an edge* belongs to that surface *if it is adjacent to one triangle of that surface. A vertex that is not on the surface is said to be an* inner vertex.

Definition 9 (Link of a vertex) *The* link *of a vertex is the set of all triangles* opposite *to that vertex in one of its adjacent tetrahedra.*

Definition 10 (Connectivity) *A set of tetrahedra is* connected through its triangles *if it is possible to link each pair of tetrahedra in the set with a set of adjacent tetrahedra of that set.*

2.2 Definition of manifold tetrahedralisations

The notion of *manifold meshes* has been mainly studied in the case of triangulated surfaces [GTLH01] but very few papers have focused on volumetric tetrahedral meshes.

Only authors that have addressed the problem of cutting into tetrahedral meshes have reported problems with preserving the manifold nature [NvdS01,BdC00] of those meshes, but without really addressing the issue.

Definition 11 (Manifold with boundaries) *A subset M of $\mathbb{R}^3$ is called a* 3-manifold with boundaries *if and only if the neighborhood of every vertex is homeomorphic either to the open ball $\mathbb{S}^2$ or to the half ball $\mathbb{S}^{2+}$ (if the point is on the surface).*

It is important to note that there is a clear distinction between the notion manifold and that of embedding : a 3-manifold mesh may not be embedded in $\mathbb{R}^3$ (if it has self intersections) and an embedded volumetric mesh in $\mathbb{R}^3$ may not be manifold.

Definition 12 (Manifold tetrahedralisation) *A topological tetrahedralisation is* manifold *if and only if the link of every inner vertex is homeomorphic to a sphere and the link of every surface vertex is homeomorphic to a half-sphere.*

Corollary 13 *According to the Euler-Poincaré formula, the number of ver-*

tices V, edges E and triangles F of the link of an inner vertex of a manifold tetrahedralisation follows the relation : $V - E + F = 2$ while those related to the link of a surface vertex is : $V - E + F = 1$

In the case of a tetrahedralisation embedded in $\mathbb{R}^3$, the conditions to obtain a manifold mesh can be simplified. Indeed, if the mesh is embedded, the link of a vertex is already homeomorphic to a sphere (otherwise, if the link of an inner vertex has non-zero genus, the mesh would not be embedded). Similarly, we can analyse the link of surface vertices in an embedded tetrahedralisation by projecting it on the unit sphere S^2. We can also define the manifold property by looking at the neighborhhod of each surface vertex :

Proposition 14 *An embedded tetrahedralisation is manifold if and only if the neighborhood, within the surface, of every surface vertex is homeomorphic to a disc.*

Indeed, the neighborhood of an inner vertex in an embedded tetrahedralisation is a closed subset of $\mathbb{R}^3$ and therefore trivially verifies the manifold property [Hof89]. Only the neighborhood of surface vertices must be checked, which can be stated as follows :

Corollary 15 *An embedded tetrahedralisation is manifold if and only if the neighborhood, within the surface, of every surface vertex is connected and if each surface edge is adjacent to exactly two surface triangles.*

Definition 16 (Singularities) *If the neighborhood of a vertex is not connected (*i.e. *if its neighborhood has several connected components), then the tetrahedralisation is said to have a* topological singularity located on that vertex. *If an edge on the mesh surface is adjacent to more than two surface triangles, then the tetrahedralisation is said to have a* singularity located on that edge.

Corollary 17 *A consequence of the above definitions is that an embedded tetrahedralisation is manifold if and only if it has no singularities located on its surface.*

3 Tetrahedron Removal Algorithm

3.1 Problem position

In the context of a laparoscopic simulator, we simulate the action of an ultrasonic lancet (cavitron) on the liver parenchyma. The liver is represented as a tetrahedralisation and the tip of the instrument as a cylinder. After perform-

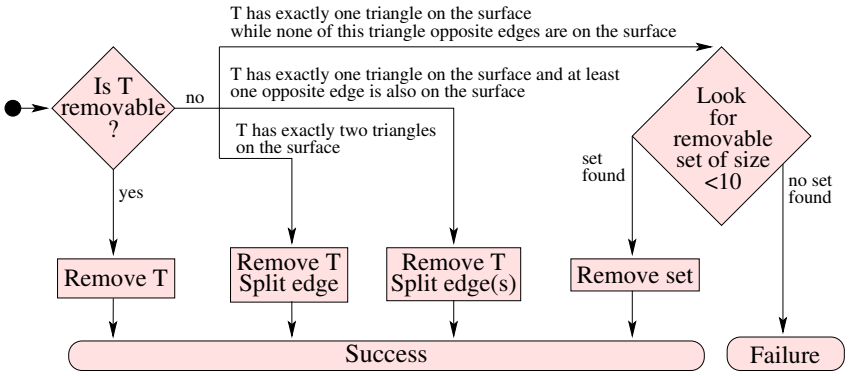


Fig. 2. Description of the tetrahedron removal algorithm.

ing a collision detection followed by a surface contact computation [FDA04], the collided tetrahedra are eventually refined and those tetrahedra still intersecting the tip of the instrument are candidates for removal. These tetrahedra are always lying on the mesh surface.

3.2 Strategy for solving topological singularities

The input of our algorithm consists in one single tetrahedron to be removed. The first step consists in testing whether or not that tetrahedron can be removed without creating any singularity. Should a singularity arise at an edge, then a simple solution based on edge splitting can be found Otherwise, the algorithm proceeds by determining a small set of tetrahedra containing the target tetrahedron, that can be removed without creating any singularity. In some cases no removable set can be found and then the removal operation is considered to be *unsuccessful* (see figure 2).

It should be emphasized that removing topological singularities in triangulations is much easier than removing singularities in tetrahedralisations. Indeed, in the former case, one can proceed by splitting the mesh around singularities at vertices and edges. This approach is not applicable in the latter case since the link of a vertex cannot be easily split between a *right* and a *left* part.

3.3 Testing removability

A basic component of our algorithm is to test whether a single tetrahedron, or a set of tetrahedra, can be removed without causing any topological problem.

3.3.1 Case of a single tetrahedron

Testing the removal of a single tetrahedron is quite straightforward. Let T be the tetrahedron to be removed:

(1) if T has no faces on the surface of the tetrahedralisation (T is then inside the mesh), it is safely removable *if and only if none of its four vertices are part of the mesh surface*. Actually we only try to remove surface tetrahedra (see part 3.1) and therefore this case will not be discussed any further;
(2) if T has exactly one triangle on the surface of the tetrahedralisation, it is safely removable *if and only if the vertex opposite to that triangle is not on the surface*. The case where that opposite vertex belongs to the surface can be decomposed into four sub-cases according to the number of edges in T, opposite to that surface triangle, that belong to the surface (see figure 3):
   b. no surface edge;
   c. exactly one surface edge;
   d. exactly two surface edges;
   e. all three edges are one the surface.

   The removal of a tetrahedron corresponding to sub-case b creates a singularity located on the vertex opposite to T surface triangle. The removal of a tetrahedron corresponding to case c, d, or e creates singularities at surface edges.
(3) if T has exactly two triangles on the surface (T makes an angle in the surface), it is safely removable if and only if the edge opposite to those two triangles does not belong to the surface (see figure 4 (a) and (b)). Otherwise the removal of the tetrahedron creates a singularity located on that edge;
(4) if T has three or four triangles on the surface (pyramid lying on the mesh or single floating tetrahedron) it can always be safely removed (see figure 4 (c) and (d)).

The only implementation issue for this test is to determine whether or not a given vertex or edge is on the mesh surface. This can be performed efficiently if all surface vertices edges and triangles are marked as such.

3.3.2 Case of a set of tetrahedra

Testing the removal of a set of tetrahedra is obviously more computationally expensive. Since we are dealing with a tetrahedralisation embedded in $\mathbb{R}^3$, we only have to check for the occurence of singularities on surface edges and vertices (see property 15). Let $\mathcal{T}$ be the set of tetrahedra to be removed. The test is performed in three stages:

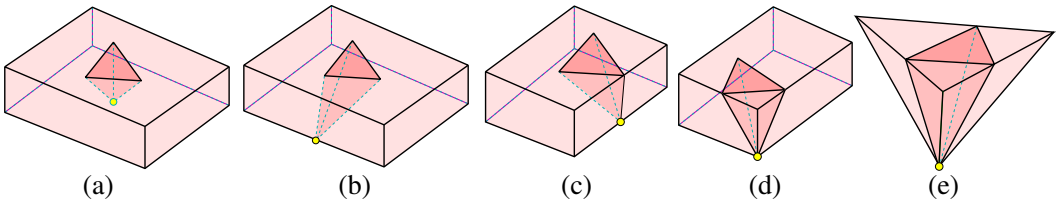


Fig. 3. Examples of tetrahedra with one single triangle belonging to the surface of the tetrahedralisation.

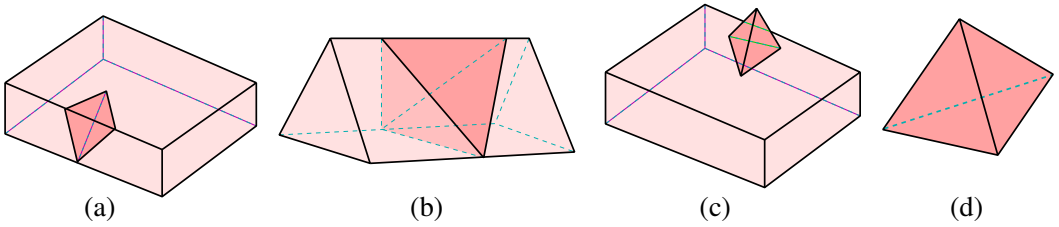


Fig. 4. Examples of tetrahedra with 2, 3 or 4 triangles belonging to the surface of the tetrahedralisation.

(1) retrieving the sets $\mathcal{T}_v$, $\mathcal{T}_e$ and $\mathcal{T}_t$ of respectively all vertices, edges and triangles lying on the surface of $\mathcal{T}$;
(2) testing each edge in $\mathcal{T}_e$ by counting the number of triangles in $\mathcal{T}_t$ adjacent to that edge;
(3) testing each vertex of $\mathcal{T}_v$ by retrieving all triangles in $\mathcal{T}_t$ adjacent to that vertex and checking if those triangles are connected by their edges.

3.4 Algorithm Description

Lets T be the tetrahedron to be removed. The outcome of the test presented in section 3.3 is one of the four cases:

- the tetrahedron is removable;
- the tetrahedron is not removable and exactly two of its triangles belong to the tetrahedralisation surface (edge singularity);
- the tetrahedron is not removable, exactly one of its triangles belongs to the tetrahedralisation surface and so does at least one of that triangle opposite edges (edge singularity-ies);
- the tetrahedron is not removable and exactly one of its triangles belongs to the tetrahedralisation surface but none of that triangle opposite edges do (vertex singularity).

If T appears to be removable it is simply removed. Otherwise we proceed according to three remaining cases.

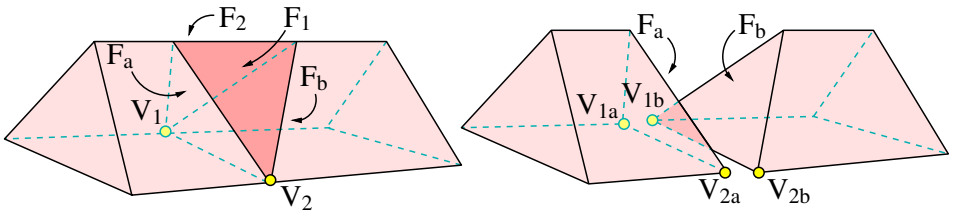
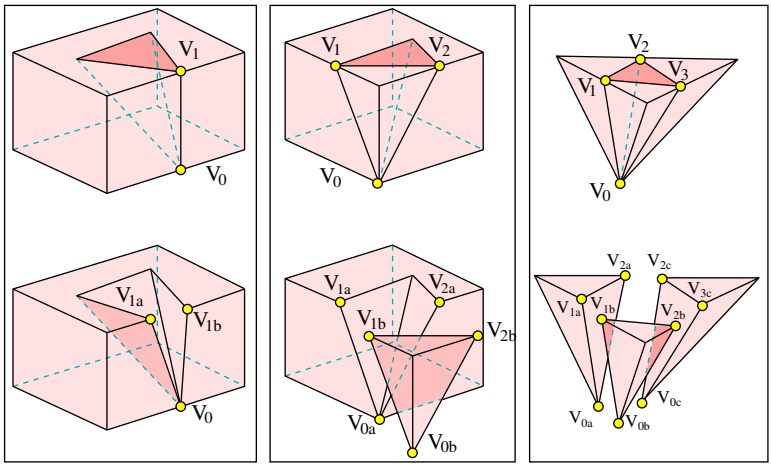


Fig. 5. Removing a singularity for a tetrahedron with exactly two triangles on the surface.



(a) (b) (c)

Fig. 6. Removing edge singularities for a tetrahedron with exactly one triangle on the surface.

3.4.1 Edge singularity: tetrahedron with exactly two surface triangles

Let T be a tetrahedron with exactly two surface triangles as shown in Figure 5. It is possible to suppress that singularity by simply splitting that edge since T divides the edge neighborhood into two connected components. The split operation then consists in separating those two connected components by creating two additional vertices and one additional edge (see figure 5).

3.4.2 Edge singularity(-ies): tetrahedron with exactly one surface triangle

Let T be a tetrahedron having exactly one surface triangle and at least one of its opposite edges on the surface (see Figure 6). Those edges divide the neighborhood of their vertices into several connected components.

a. if exactly one edge of T adjacent to that vertex belongs to the surface, then the singularity can be suppressed by splitting the other vertex of that edge;
b. if exactly two edges of T adjacent to that vertex belong to the surface,

then the singularity can be suppressed by splitting those two edges;
c. if the three edges of T adjacent to that vertex belong to the surface, then that singularity can be suppressed by splitting those three edges.

3.4.3 Vertex singularity: tetrahedron with exactly one surface triangle

Let T be a tetrahedron having exactly one surface triangle but with none of its opposite edges on the surface (see Figure 8). Then, there is no simple splitting operation possible to suppress the singularity. Our strategy is to determine a set $\mathcal{T}_{removable}$ containing T that can be removed. There are obviously many possible removable sets, the most trivial one being the whole mesh. Therefore, we need to give additional criteria for defining the optimal set $\mathcal{T}_{removable}$:

- minimize the size of $\mathcal{T}_{removable}$, i.e. the number of tetrahedra to remove. Indeed, it is very reasonable to restrict as much as possible the side effect of preserving the manifold nature of the mesh. In the context of surgery simulation, it simply avoids to remove a large volume each time the action of the ultrasonic lancet is simulated. Note that we could have replaced this criterion by minimizing the overall volume of $\mathcal{T}_{removable}$ rather than its size;
- restrict the tetrahedra of $\mathcal{T}_{removable}$ to be adjacent to at least one of the four vertices of T. The motivations are twofold. First, we want $\mathcal{T}_{removable}$ to be located around T to enforce the visual realism of the cutting. Second, we want to limit the computation time of $\mathcal{T}_{removable}$ since we are targeting real-time applications.

Unfortunately, limiting spatially the search of $\mathcal{T}_{removable}$ implies in some cases the absence of solutions. Figure 7 shows an example in 2D for such case. For tetrahedralisation, the occurence of such cases is very rare. When the search fails, an empty set is outputted $\mathcal{T}_{removable} = \{\}$.

Instead of testing blindly for a removable set, it is worth looking at the link of the problematic vertex (see figure 8). Figure 9(a) shows a typical flattened map of the link at such a vertex. The removal of target tetrahedron would create a hole in the vertex link which will no longer be homeomorphic to a half sphere. Our approach is to remove that additional hole by searching a path starting from the removed tetrahedron and moving towards the border of the vertex link. This search is performed in a breadth-first manner in order to get the shortest path and is further arbitrarily limited to a depth of 10 in order to restrict the computational time. When a path is determined, the removal of the corresponding set is tested and, upon success that set $\mathcal{T}_{removable}$ is given as a result of the algorithm.

If the test fails, we use an additional heuristics. It appears to be often the case that the path found divides the adjacency of one of the tetrahedron vertices into two connected components (see figure 9(b)). Therefore, we also test

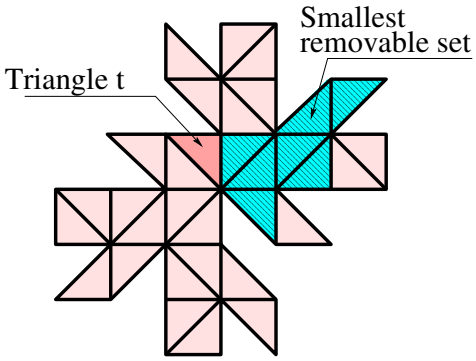


Fig. 7. Example of a manifold triangulation where it is not possible to find a set of triangles adjacent to triangle t, that can be removed without creating any singularities.

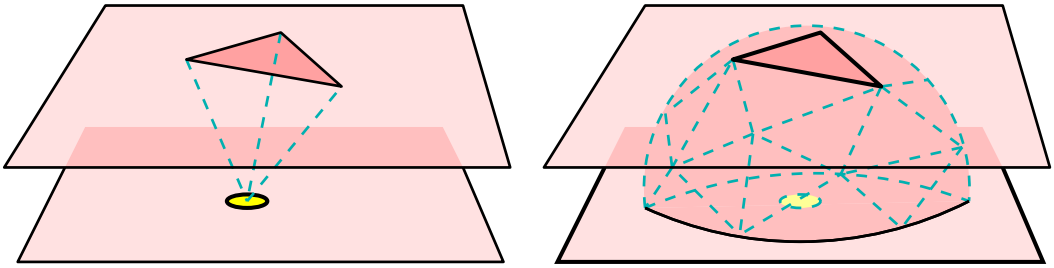
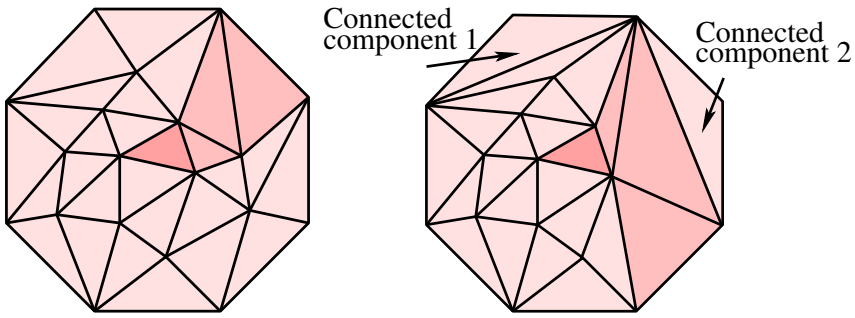


Fig. 8. (Left) A tetrahedron with a single triangle on the mesh surface and having its opposite vertex also on the surface. (Right) The link of that vertex.



(a) (b)

Fig. 9. (a) Flattened map of the link of the vertex displayed in figure 8. Because of the manifold property of the mesh, we know that the link of a vertex that lies on the surface is homeomorphic to a half sphere and therefore can be flattened. The border edges correspond to the edges opposite to the vertex in the mesh surface (the bottom of the half sphere on figure 8). (b) example of a path dividing a vertex link in two connected components.

the removability of the given path extended with each of the two connected components. If one of that two sets appears to be actually removable it is temporarily stored and will be eventually used if we no smaller set is determined later on. If no paths nor extended paths are found then we output an empty set $\mathcal{T}_{removable} = \{\}$.

Note that one could always remove tetrahedron T, by refining the singular vertex and by removing both the refined tetrahedron and the refined vertex adjacency (see [FDA02] for more details). However, in practise, we chose not to use such algorithm because it creates ill-shaped tetrahedra of small size. Instead, we found that if tetrahedron T cannot be removed at a given time t, it is very likely to become removable later on, once a neighboring tetrahedron has been removed. This empirical property is very important in the context of surgical simulation and explains why the user can always resect the selected piece of tissue.

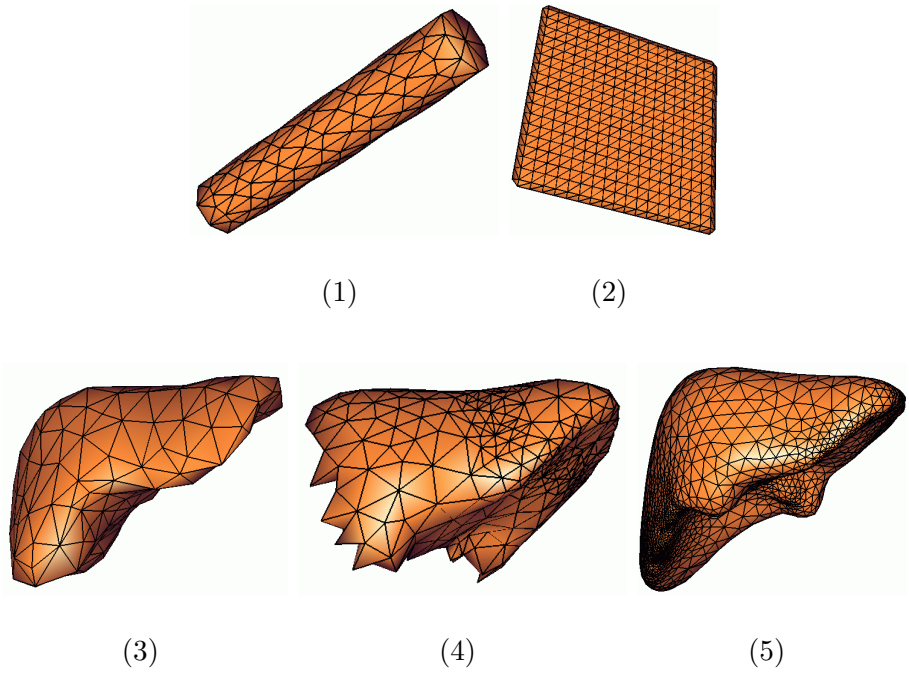


Fig. 10. Five tetrahedral meshes used for the tests.

4 Evaluation of the algorithm

4.1 Descriptions of the tests

In order to evaluate the performance and suitability of this algorithm, we have removed one-by-one several tetrahedra from different meshes and evaluated the percentage of resolved cases and the average size of the removed sets. To be more significant, if a non-removable tetrahedron is encountered several times it is taken into account only once.

The different meshes are (see figure 10):

(1) a simple cylinder;
(2) a thin sheet with an average thickness of two tetrahedra;
(3) a coarse mesh of a liver (3970 tetrahedra);
(4) a locally refined mesh of the same liver (8454 tetrahedra);
(5) a locally refined mesh of a liver (9933 tetrahedra for a third of the liver) obtained by applying an edge split algorithm.

The result appears to be quite stable from one mesh to the other independently of its number of tetrahedra removed, of its geometry or of the quality of its tetrahedra..

4.2 Occurrence of topological singularities

Mesh	number of operations	% of removable tetrahedra	% of singularities	
			on edges	on vertices
1	278	83	14	3
2	358	72	24	4
3	418	81	16	3
4	693	83	14	3
5	1881	83	15	2

The table above lists for each type of mesh the occurence of edge and vertex singularities when removing a tetrahedron lying on the mesh surface. The three leftmost figures sum up to 100%. Except for the thin sheet the ratio of non removable tetrahedra is nearly 20%. However, one would only remove those tetrahedra than do not create singularities, then the occurence of non-removable tetrahedra would drastically increase until no surface tetrahedra can be removed any longer. This shows that if the manifold nature of the mesh is to be enforced, it is mandatory to cope with the presence of singularities as proposed in the algorithm described above.

4.3 Performance for vertex singularities deletion

The majority (80%) of the topological problems encountered are caused by edge singularities and can be solved with the simple splitting method presented in section 3.4. For the remaining 20% we propose to determined a small removable set as described at the end of section 3.4. Globally that method is rather efficient since it solves more than 97% of the cases. Note also efficiency

of the *extended path heuristics* that solves most of the remaining pathological cases.

Mesh	% of solved cases		% of unsolved cases	average size of removed set		
	simple path	extended		simple	extended	global
1	100	0	0	4.7	-	4.7
2	94	6	0	4.3	4.0	4.3
3	84	16	0	4.6	5.0	4.7
4	78	16	6	6.3	4.3	6.0
5	86	7	7	5.5	8.6	5.7

4.4 Evaluation of the strategy

Mesh	General		Problematic tetrahedra	
	% of removal unsolved	average size of removed set	% of removal unsolved	average size of removed set
1	0	1.1	0	1.4
2	0	1.1	1.0	1.3
3	0	1.1	2.5	1.4
4	0.15	1.1	3.4	1.5
5	0.15	1.1	1.8	1.4

This table lists the percentage of cases solved and the average number of tetrahedra removed during one operation. Almost 99.8% of the tetrahedra can be removed successfully. We consider that an average of 0.2% of unremovable tetrahedra is a rather good result for a cutting strategy. Furthermore, when simulating the action of an ultrasonic lancet, the typical number of tetrahedra to be removed in a single time step stands between 4 and 6, and therefore the probability to end-up with no removable tetrahedra decreases exponentially.

4.5 Real-Time Computation

The proposed algorithm is largely compatible with real-time requirements. Indeed, the average time for removing one tetrahedron (in non trivial cases)

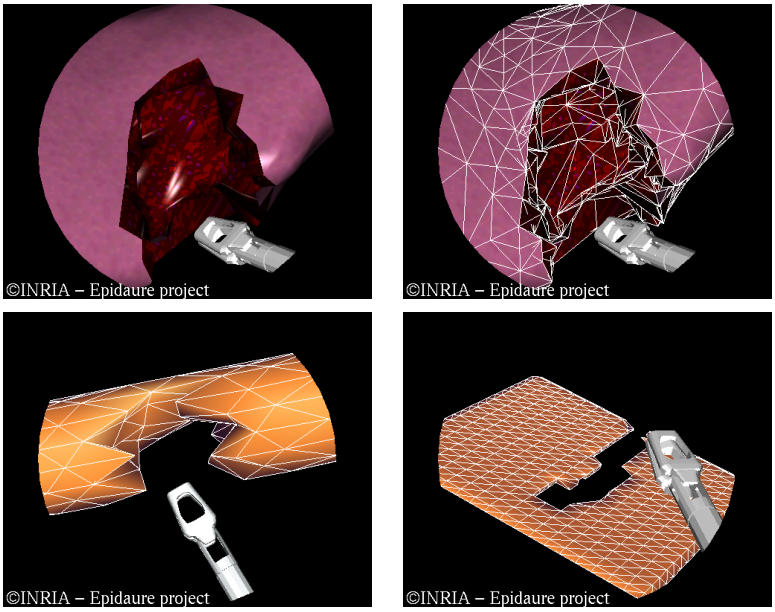


Fig. 11. Examples of volumetric cuts performed with our tetrahedron removal algorithm.

is less than 0.2 millisecond on a Pentium III computer. Unsurprisingly, the worst case is reached when the search of a removable set is performed with the maximum depth. For a depth of 10 tetrahedra, the algorithm can take up to 5 ms whereas with a depth of 8, it goes down to 3 ms. There is therefore a trade-off between the efficiency of the algorithm and its maximum time of computation.

5 Conclusion

We have presented a tetrahedron removal algorithm that guarantees the preservation of the manifold nature of a tetrahedral mesh. This method, rather simple to implement, does not significantly worsen the geometric quality of tetrahedra. Furthermore it is a rather efficient algorithm since it can solve almost 99.8% of all requested removal.

This algorithm has been implemented in an hepatectomy simulator, providing an acceptable solution for resecting the hepatic parenchyma. We believe the proposed algorithm can still be improved, for instance, by finding additional alternatives to the breadth-first search described in section 3.4.3. In some configurations where several edges are part of the surface, a vertex split-based solution could be advantageous if one does not have to deal simultaneously with several vertex singularities. Also, testing for a removable set of tetrahedra might be improved by only looking at the link of edges or vertices where singularities occur. Another field of investigation is the application of the “refine

and remove" strategy to obtain finer and more regular cut surfaces.

6 Bibliography

References

[BdC00] François Boux de Casson. *Simulation dynamique de corps biologiques et changements de topologie interactifs.* Thèse de science, Université de Savoie, Chambéry (FR), December 2000.

[BG00] Daniel Bielser and Markus H. Gross. Interactive simulation of surgical cuts. In *Proc. Pacific Graphics 2000*, pages 116–125. IEEE Computer Society Press, October 2000.

[BSM+02] C Bruyns, S Senger, A Menon, S Wildermuth, K Montgomery, and R. Boyle. Survey of interactive mesh cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *Journal of Visualization and Computer Animation*, 13(1), 2002.

[FDA02] Clément Forest, Hervé Delingette, and Nicholas Ayache. Cutting simulation of manifold volumetric meshes. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2488 of *LNCS*, pages 235–244, Tokyo, September 2002. Springer.

[FDA04] Clément Forest, Hervé Delingette, and Nicholas Ayache. Surface contact and reaction force models for laparoscopic simulation. In *International Symposium on Medical Simulation*, June 2004.

[GTLH01] André Guéziec, Gabriel Taubin, Francis Lazarus, and Bill Horn. Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Trans. on Visualization and Computer Graphics*, pages 136–151, 2001.

[Hof89] Christoph M. Hoffmann. *Geometric and solid modeling: an introduction.* Morgan Kaufmann Publishers Inc., 1989.

[MK00] Andrew B. Mor and Takeo Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *MICCAI*, pages 598–607, 2000.

[NvdS01] Han-Wen Nienhuys and A. Frank van der Stappen. Supporting cuts and finite element deformation in interactive surgery simulation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'01)*, pages 145–152. Springer, October 2001.

[NvdS02] Han-Wen Nienhuys and A. Frank van der Stappen. Interactive cutting in triangulated surfaces, a delaunay approach. In *Procs. of the*

Fifth International Workshop on Algorithmic Foundations of Robotics (WAFR'02), December 2002.

[PBKD02] Celine Paloc, Fernando Bello, Richard I. Kitney, and Ara Darzi. Online multiresolution volumetric mass spring model for real time soft tissue deformation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2489 of *LNCS*, pages 219–226. Springer, September 2002.

[PS03] Giuseppe Patanè and Michela Spagnuolo. Triangle mesh duality: Reconstruction and smoothing. In *Mathematics of Surfaces: 10th IMA International Conference*, volume 2768 of *LNCS*, pages 111–128. Springer-Verlag, November 2003.

[Qin00] Hong Qin. Fem-based dynamic subdivision surfaces. *8th Pacific Conference on Computer Graphics and Applications*, pages 184–191, October 2000. ISBN 0-7695-0868-5.

[SHS01] D. Serby, M. Harders, and G. Székely. A new approach to cutting into finite element models. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'01)*, pages 425–433. Springer Verlag, October 2001.

[VHM+99] Gerrit VoSS, James K. Hahn, Wolfgang Muller, , and Robert W. Lindeman. Virtual cutting of anatomical structures. In *Medicine Meets Virtual Reality (MMVR '99)*. IOS Press, January 1999.

[VPBM01] A. Vlachos, J. Peters, C. Boyd, and J. Mitchell. Curved PN triangles. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, August 2001.