Hybrid Cutting of Deformable Solids

Denis Steinemann*
Computer Graphics Laboratory
ETH Zurich

Matthias Harders†
Computer Vision Laboratory
ETH Zurich

Markus Gross‡
Computer Graphics Laboratory
ETH Zurich

Gabor Szekely§
Computer Vision Laboratory
ETH Zurich

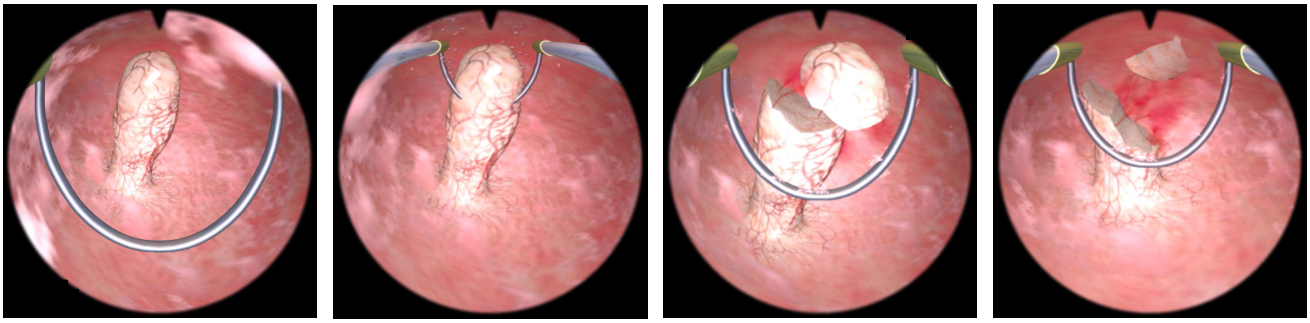


Figure 1: Ablating a polyp in the hysteroscopy simulator.

ABSTRACT

A central training objective of virtual reality based surgical simulation is the removal of pathologic tissue. This necessitates stable, real-time updates of the underlying mesh representation. Within the framework of a hysteroscopy simulator, we have developed a hybrid cutting approach for tetrahedral meshes. It combines the topological update by subdivision with adjustments of the existing topology. Moreover, the mechanical and the visual model are decoupled, thus allowing different resolutions for the underlying mesh representations. With our method, we can closely approximate an arbitrary, user-defined cut surface while avoiding the creation of small or badly shaped elements, thus strongly reducing stability problems in the subsequent deformation computation. The presented approach has been integrated into a virtual reality training system for hysteroscopic interventions. The performance of the algorithm is demonstrated by examples of intra-uterine tumor ablations.

Keywords: tetrahedral meshes, cutting, surgical simulation, mass-spring models

1 INTRODUCTION

Surgical simulation has become a popular application in the field of virtual reality based training [11]. One of the essential parts of such a system is the cutting of soft, deformable tissue. Since tetrahedral meshes are a popular representation for volumetric objects, methods to impose topological changes on such meshes are required. Cutting a tetrahedral mesh is a non-trivial problem, due to several conflicting requirements. On the one hand, the cutting process should not create badly shaped elements, which could cause numerical instabilities during deformation calculation. On the other hand, the user defined cut trajectory should be closely approximated for realistic appearance. So far, most methods have concentrated only on one of these problems. We take a different approach by combining these methods to develop a stable and realistic cutting system. We introduce a hybrid approach for cutting tetrahedral meshes, which approximates an incision as well as possible, while avoiding the creation of small or degenerate tetrahedral elements. Moreover, we decouple the mechanical simulation and visualization domains, which allows modelling of fine surface detail without increasing the computational burden on the physical simulation.

The paper is organized as follows: in section 3, we briefly review previous work related to cutting tetrahedral meshes in virtual surgery. Thereafter, section 4 presents our hybrid cutting approach. Section 5 deals with the visual representation and its coupling to the deformation model. Experimental results are discussed in section 6, and we conclude and provide an outlook to future work in section 7.

2 MEDICAL APPLICATION AREA

We have developed our new cutting approach in the context of hysteroscopy simulation. In hysteroscopy, a surgeon uses a curved loop electrode as a cutting tool to ablate pathological tissue inside the uterus (Figure 1). The loop electrode is positioned behind the pathology and then advanced towards the camera from the back, cutting off tissue parts. Due to this, the actual cutting process cannot be seen by the surgeon. For this reason, non-progressive cutting, where a tetrahedral element is decomposed only once it has been completely traversed, is a reasonable approximation for our application area, and so we define the cut only once the loop electrode has traversed the pathology. Moreover, there is little, if at all, resistance to the cut tool movement through the tissue. Therefore, in the current stage, we do not model any interaction of the cutting tool with the deformable object during a cut. The deformable objects in our simulation are represented by tetrahedral meshes and simulated by a mass-spring model, including distance-, volume- and surface-preserving forces [19].

*e-mail: denis.steinemann@inf.ethz.ch
†e-mail: mharders@vision.ee.ethz.ch
‡e-mail: grossm@inf.ethz.ch
§e-mail: szekely@vision.ee.ethz.ch

3 Related Work

Previous approaches can be classified into three different categories. The first and simplest methods delete tetrahedra which are touched by the cutting tool, as for instance described in [7, 4]. No new elements are created, however, this approach violates the physical principle of mass conservation. Furthermore, it requires very high resolution meshes for acceptable visual quality.

The second class of methods restricts incisions to be aligned with existing faces. This has the advantage of only small increase in element count, even after multiple cuts. However, the approximation quality of the cut path as well as the smallest possible size of excised material pieces depends highly on the initial resolution of the mesh. [15] and [18] apply the concept of node snapping, where existing nodes are moved to the cut surface to approximate the tool trajectory. However, this requires an update of mesh parameters of the undeformed mechanical model, which can be difficult if displacements are large.

In the third class of methods, elements are actually subdivided into smaller ones. Since a mesh-based physical simulation needs a consistent mesh at all times, a cut tetrahedron must be decomposed into smaller elements such that a new consistent configuration results. To achieve this, many small tetrahedra have to be created, which dramatically increases element count and thus may slow down the simulation substantially. Even worse, very small or badly shaped tetrahedra (i.e. slivers) may be created, causing simulation instabilities, unless very small time steps are used. In [10] the problem of badly-shaped tetrahedra is handled by removing such elements on-the-fly. However, not all elements can be removed without larger changes to the mesh. Moreover, the approach again violates the principle of mass conservation.

There is a large number of ways a tetrahedron can be cut, depending on the number of edge intersections, and each must be handled separately. For each case, pre-defined decomposition schemes are usually used to subdivide tetrahedra [8, 9, 3]. In addition, great care must be taken to ensure that new adjacency information is always correct.

Most existing cutting approaches are either non-progressive or semi-progressive meaning that a tetrahedron can only be cut once the cutting tool has completely moved through the element. Depending on mesh resolution, this can cause a noticeable lag between the actual cutting and the movement of the tool, making it difficult to control the cut. In [1] a progressive approach has been presented, where the decomposition of a tetrahedron is changed depending on the movement of the cutting tool inside an element. However, the approach is highly non-trivial to implement and also poses some stability problems due to badly-shaped elements. Progressive mesh cutting has also been described in [13], but due to badly-shaped elements small time steps are required, which prevents the simulation from running in real-time.

Finally, work has been published where simulation and visualization domains are decoupled. [12] presented a virtual node algorithm, where nodes and elements are copied such that no new smaller elements are created. Elements are only decomposed in the visualization domain. A tetrahedron cannot be cut more than three times, however, and the resolution of the surface depends on the resolution of the underlying tetrahedral mesh. [14] simulates elasticity and plasticity as well as fracture on a low resolution tetrahedral mesh, while using an embedded triangle mesh to visualize the object surface. However, the new visualization surfaces created after an object is fractured depend on the resolution of the tetrahedral mesh. Other related work focusing on fracture modelling is reported in [17]. They use a related approach to perform offline computations of fracture propagation in tetrahedral FEM meshes.

4 Cutting Tetrahedral Meshes

4.1 Overview

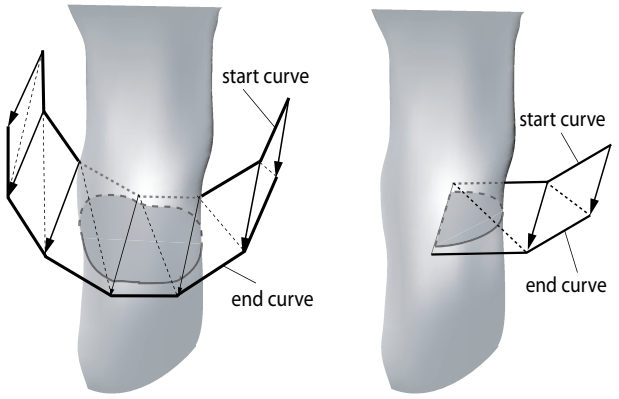
We introduce a new hybrid approach to cutting tetrahedral meshes. A given cut trajectory - in the following called *sweep-surface* - is approximated as closely as possible, while lowering the increase in element count and avoiding the creation of small and badly-shaped tetrahedral elements. When cutting a deformable tetrahedral mesh, we carry out the following steps:

1. Define sweep-surface according to tool movement.
2. Determine edges intersected by the sweep-surface.
3. To avoid small or badly shaped elements, identify the nodes close to the sweep surface. Along these nodes the mesh can be separated.
4. Separate existing tetrahedra or decompose them into smaller sub-elements.
5. Move selected nodes onto the sweep-surface for better cut approximation (node-snapping).
6. Improve mesh quality with local relaxation.
7. Update deformation parameters for stable simulation.

4.2 Definition of the Sweep-Surface

Before a cut can be performed, the sweep-surface must be defined. It can be any surface that does not self-intersect and can be triangulated. Moreover, it can intersect the model totally or only partially.

In our implementation, the cutting tool is defined as a curve approximated by n line segments, as can be seen in Figure 2. The system constantly checks for collisions of the cutting tool with the deformable model. If this has been the case, the starting curve of the sweep-surface is defined. Thereafter, the tool movement is tracked until it fully leaves the deformable object. At this point, the end



(a) Sweep-surface defined by start and end curve representing cutting tool.

(b) A partial cut.

Figure 2: Definition of the sweep-surface: (a) The cutting tool is defined as a curve defined by n line segments. Once the tool collides with the model, the start curve of the sweep-surface is defined. When the tool leaves the object, the end curve is defined. Connecting corresponding sample points of both curves linearly interpolates the cut trajectory and defines the sweep-surface, which can be triangulated easily. (b) In a partial cut the sweep-surface does not intersect the model totally.

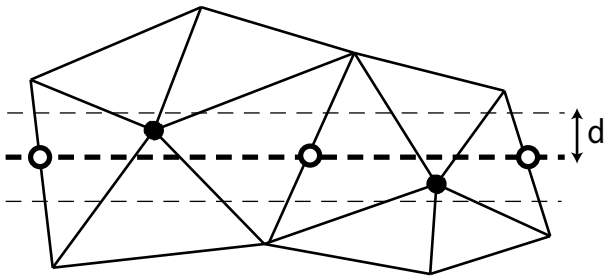
curve of the sweep-surface is defined. The cut trajectory is then linearly interpolated by connecting corresponding sample points on the start and end curves. Now, the surface can easily be triangulated by $2n$ triangles.

4.3 Selecting Cut-Nodes and Cut-Edges

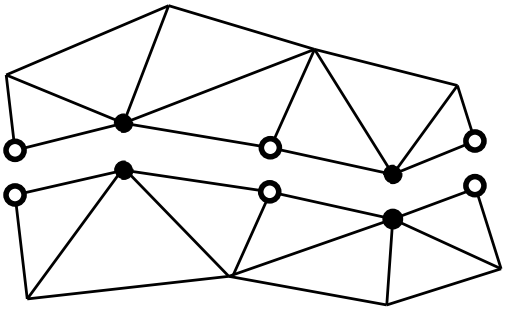
Once the sweep-surface has been defined, it is possible to determine those tetrahedra which are simply separated from each other and those that are actually decomposed into smaller sub-elements. If an edge is cut close to a node, the cut is constrained to go through that node, thus avoiding the generation of small tetrahedra and an increase in element count. If an edge is incised close to the middle, chances are lower that small sub-elements are created. Thus, such an edge is a candidate for subdivision. For the discussion below, we introduce the following terms:

cut-node: a node that is sufficiently close to the sweep-surface and to which a cut is constrained to. None of its incident edges can be cut.

cut-edge: an edge which is intersected by the sweep-surface and for which none of its end nodes are cut-nodes.



(a) For all edges which are intersected by the sweep-surface, the end node closer to the intersection point is selected. If this node lies within a threshold d, it is marked as a cut-node. All intersected edges which have no cut end nodes are marked as cut-edges.



(b) All cut-nodes are duplicated and all cut-edges are split. The sweep-surface is closely approximated without creating ill-shaped elements.

Figure 3: Selecting cut nodes in 2-D.

To find the sets of cut-nodes (N_c) and cut-edges (E_c), we have implemented a similar approach to [15]. We start out with E_c containing all edges e, which intersect the sweep-surface. For all edges in E_c, the end node n which lies closest to the intersection point is selected. As shown in Figure 3, if the node's distance to the sweep-surface is smaller than a threshold d, the node is added to N_c, and all intersected edges incident to n are removed from E_c. Thereafter, all cut-nodes in N_c are duplicated, while all cut-edges in E_c are split into two new edges, thereby also creating two new collocated end nodes at position $\mathbf{p}_{int}$. The reference position of these new nodes in the undeformed state, $\mathbf{p}^0_{int}$, is obtained by linearly interpolating the reference positions $\mathbf{p}^0_0$ and $\mathbf{p}^0_1$ of the previous end nodes of the cut-edge, using the intersection parameter t in the deformed configuration.

$$\begin{aligned} t &= ||(\mathbf{p}_{int} - \mathbf{p}_0)||/||(\mathbf{p}_1 - \mathbf{p}_0)|| \\ \mathbf{p}^0_{int} &= \mathbf{p}^0_0 + t(\mathbf{p}^0_1 - \mathbf{p}^0_0) \end{aligned} \quad (1)$$

In rare cases an edge may be cut twice, because the sweep-surface can be curved. If this happens, both cuts are simply discarded and corresponding tetrahedra are decomposed according to the cuts through its other edges as described below.

4.4 Hybrid Tetrahedron Decomposition

Now that the sets of cut-nodes and cut-edges have been determined, we have all the pre-requisites to actually realize the cut by splitting and decomposing tetrahedra.

We have extended the methods reported in [3, 8], which restricted possible cut states to intersections with one to four edges of a tetrahedron. This resulted in five distinct, rotationally-invariant decomposition schemes (A-E) as depicted in Figure 4. We introduce three new schemes, where a tetrahedron has either one cut-node and two cut-edges (scheme X), two cut-nodes and one cut-edge (Y), or only three cut-nodes (Z). In these schemes, the cut is forced to go through nodes, avoiding the creation of small or badly-shaped tetrahedra and lowering the increase in element count.

The decomposition schemes (A-E) can be characterized by a simple binary six-digit cut-edge-code, where each digit stands for a specific edge being cut or not. Meanwhile, schemes X and Y additionally need a binary four-digit cut-node-code to determine which of its nodes are cut. We restrict the schemes with cut-nodes to complete cuts only, we do not cut nodes for tetrahedra which are only partially cut by the sweep-surface. Therefore, a node which could be marked as a cut-node is always treated as a normal, non-cut node, if one of its incident tetrahedra is only partially cut. After computing the cut-edge-code and cut-node-code for each tetrahedron, the mesh can be separated along the sweep-surface. All tetrahedra which have one or two cut-nodes and at least one cut-edge are decomposed according to schemes X and Y. If a tetrahedron has only cut-nodes but no cut- edges, as in scheme Z, it is simply detached from its neighbors. All tetrahedra containing only cut-edges but no cut-nodes are decomposed according to schemes A-E.

Furthermore, we apply a new symmetric face triangulation scheme such that the sub-elements have a better shape. In Figure 5(a), the face triangulation scheme suggested in [3] can be seen, while Figure 5(b) shows our approach. Experiments have shown that with our technique we can use larger time steps because elements are less likely to be badly-shaped and element sizes are more equally distributed. The (asymmetric) decomposition schemes from [8] are more optimal in terms of element count, but for each scheme, several sub-cases must be implemented so that faces shared by two tetrahedra are triangulated consistently.

The reference (and similarly the deformed) position of a new face node is computed by taking the average of the reference positions of the four corner nodes of the face prism.

4.5 Node Snapping

Along cut-nodes, the mesh is not aligned with the cut trajectory. Although cut-nodes are always close to the sweep-surface and therefore the mesh usually does not look too jagged, it can still be desirable to have the mesh perfectly aligned to the sweep-surface. Therefore, a cut-node n is relocated by projecting it orthogonally onto the given sweep-surface, yielding a new position $\mathbf{p}_{new}$. Since this is done in the deformed configuration, the new reference position $\mathbf{p}^0_{new}$ of the node n must be computed as well. This is done

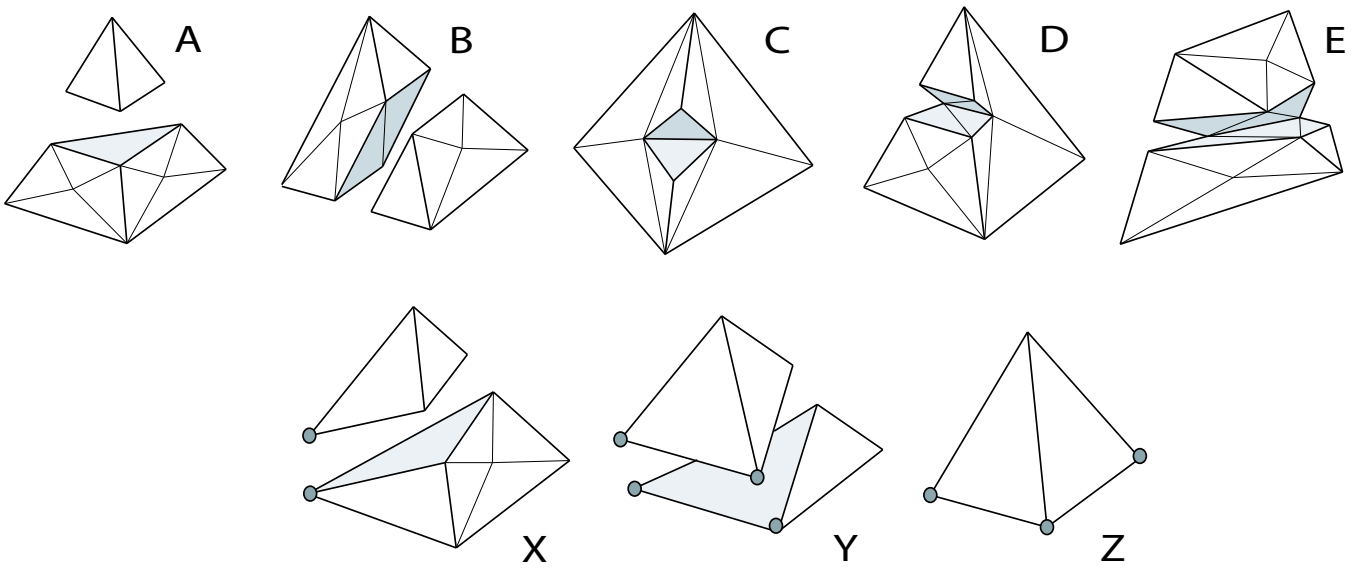
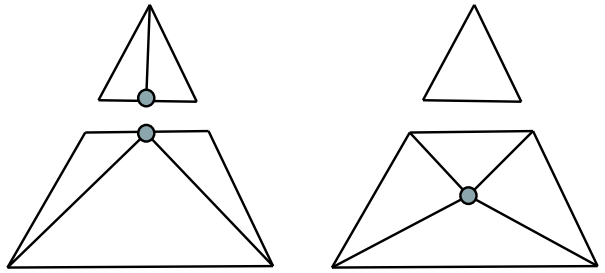


Figure 4: There are five rotationally-invariant schemes to decompose a tetrahedron (A-E) which has at least one of its edges cut. Additionally, three new schemes are added, where one, two or three existing nodes are separated (X-Z). This helps avoiding small tetrahedra created when using decomposition schemes A-E for cuts close to a node.



(a) Symmetric triangulation of a tetrahedral face according to [3].

(b) Our face triangulation approach, where sub-elements are better shaped and have more equal sizes.

Figure 5: Enhanced symmetric triangulation of a tetrahedron face.

similar to [16]. Barycentric coordinates of $\mathbf{p}_{new}$ with respect to a tetrahedron T incident to n are computed in the deformed configuration. Using these coordinates and the reference positions of the nodes of T, $\mathbf{p}^0_{new}$ is computed.

Because all cut-nodes are within a small threshold distance d to the sweep-surface, their displacement is small compared to the size of tetrahedral elements. Therefore, updating mesh parameters is easier and more stable than with previous methods [15, 18]. However, snapping nodes to the sweep-surface can create badly-shaped or even degenerate tetrahedra. In [10], these tetrahedra are removed in an extra step. In our implementation, we do not snap nodes that lead to degenerate tetrahedra.

4.6 Mesh Relaxation

Even with our hybrid cutting approach and improved decomposition schemes, it is possible that elements with very different sizes may be created, decreasing the stability of the simulation. Therefore, to further improve mesh quality after a cut, we perform a sequence of mesh relaxation steps, similar to the 2D method described in [18]. First, we define a set S containing all nodes n in the 2- or 3-step-neighborhood of a cut. To preserve the shape of the model, only interior nodes can move in all directions. Nodes on the sweep surface can move within the surface, while those on the original surface or surfaces resulting from an earlier cut are fixed. All edges incident to a movable node from S are now considered as springs whose rest lengths equal the average length of these edges. All nodes are considered as mass points and have equal masses. To enforce more even spacing between nodes, we apply a force resulting from the Lennard-Jones potential energy function.

The movement of a node n_i is governed by the following equation of motion:

$$m_i\ddot{\mathbf{x}}_i + c_i\dot{\mathbf{x}}_i = (w_{spring}\sum_j \mathbf{F}^{spring}_{ij} + w_{particle}\sum_j \mathbf{F}^{LJ}_{ij}) \qquad (2)$$

where $\mathbf{F}^{spring}_{ij}$ and $\mathbf{F}^{LJ}_{ij}$ are the spring and Lennard-Jones forces, respectively, between node n_i and its neighbor n_j. w_{spring} and $w_{particle}$ represent weighting constants.

Explicit integration is applied to solve the above equation. Due to high damping, overall computation time is only slightly increased, and oscillations are avoided. We perform several mesh relaxation steps to obtain a sufficiently good mesh quality. New rest positions for relaxed nodes are computed in the same way as for node snapping (section 4.5).

It must be noted that due to the constraint of fixing surface nodes of the tetrahedral mesh, mesh relaxation cannot improve mesh quality greatly. The presented hybrid tetrahedral decomposition schemes play a much more important role in the improved stability of the simulation. However, if a larger number of internal tetrahedra would be cut, the effect of the relaxation step would be more prominent.

4.7 Adaptation of Masses and Reference Values

After a cut has been performed, nodes have been snapped and the mesh relaxation has improved the quality of new tetrahedra, we must update the rest quantities of edges and tetrahedra correctly to keep the underlying deformation simulation stable.

To compute rest edge lengths and rest tetrahedron volumes, we simply use the reference positions of corresponding nodes, which

have been computed during the cutting process (section 4.3), in node snapping and mesh relaxation. The masses of the simulation nodes which have at least one incident tetrahedron that was affected by the cut must also be updated. This is because after a cut, smaller tetrahedra appear and volumes and edge lengths may have changed. The mass of a simulation node n_i is proportional to the rest volumes V_0 of incident tetrahedra j, which have a density k:

$$m_i = k \sum_j \frac{V_j^0}{4} \tag{3}$$

This guarantees mass conservation, since the total volume is not changed by a cut. Stiffness constants k_s^i and k_v^i for spring and volume-preservation forces are adapted to [19]:

$$\begin{aligned} k_s^i &= \frac{k(m_1+m_2)}{(l_i^0)^2} \\ k_v^i &= \frac{k(m_1+m_2+m_3+m_4)}{(v_i^0)^2} \end{aligned} \tag{4}$$

5 Cutting the Visualization Surface

In most previous cutting approaches, the simulation and visualization domains were strongly coupled. In order to represent intricate surface details, high-resolution tetrahedral meshes are needed, which substantially slows down a simulation. In approaches where cuts can be performed only along existing edges and faces, the size of the smallest portions of material that can be cut off depends highly on the resolution of the mesh. To alleviate these problems, we have decoupled the simulation and visualization domains into a low-resolution tetrahedral mesh for physical simulation and an embedded high-resolution (closed) triangle mesh for surface rendering. The embedded surface is animated together with the tetrahedral mesh. We build on the approach of [14]. For each surface vertex, the closest tetrahedron is found and barycentric coordinates of the vertex with respect to the linked tetrahedron are computed and stored. When the tetrahedral mesh deforms, the position of each vertex is interpolated using the linked tetrahedron and the stored barycentric coordinates.

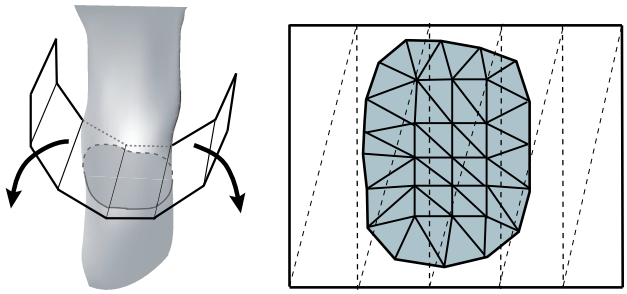
5.1 Incision into the Surface Mesh

Cutting the two-dimensional surface mesh is similar to the three-dimensional case. First, the sweep-surface must be defined (refer to section 4.2). Then, each triangle can be decomposed according to two rotationally invariant schemes (two edges cut for a complete cut, one edge cut for a partial cut). We do not cut vertices in the surface mesh, since we do not care about badly-shaped triangles in the visualization surface.

As in the 3D-case, the reference position of new vertices created when an edge is cut is interpolated from the reference positions of the end vertices of the edge and the intersection parameter t. Texture coordinates for the new vertices are computed similarly. Using this reference position, a linked tetrahedron is found and barycentric coordinates are computed. For old surface vertices whose linked tetrahedron has been decomposed or now lies on the other side of the sweep-surface, a totally new tetrahedron must be determined. For old surface vertices whose linked tetrahedron had only one or more nodes snapped, but was not decomposed, only the barycentric coordinates must be recomputed. In all cases, we must make sure that both the tetrahedron and the surface vertex lie on the same side of the sweep-surface.

5.2 Generating New Surfaces

After the original surface mesh has been separated, two additional interior cut surfaces have to be created. Generating these new surfaces consists of two steps: first, we must find the boundary of the new surface, the cut-boundary. Then, using the cut-boundary and the given sweep surface, it is triangulated. Note that this triangulation is totally independent of the volumetric mesh, contrary to [14], where the new surface is aligned to the (coarse) tetrahedral mesh.



(a) The cut boundary is determined. It lies on the sweep-surface and is mapped to a planar 2D surface according to its parameterization.

(b) The new surface is triangulated in 2-D, using the mapped cut boundary. Finally, the new surface is mapped back into 3-D.

Figure 6: Triangulation of the new surfaces.

To find the cut-boundary, we start at a new surface vertex (i.e. a vertex created when an edge is cut) and traverse along the boundary, which is defined by edges which have exactly one adjacent triangle, until the start vertex is visited again. It must be noted that this approach works only unconditionally if the original mesh is closed. Also, if the model is concave, there may be more than one closed cut-boundary curve. In case of a partial cut as shown in Figure 7, we must first determine the two face vertices, which are defined as the two intersection points of the sweep-surface boundary with the surface mesh. We assume that there are exactly two such face vertices, which is the case for partial cuts through convex, closed surfaces. The face vertices can be found directly during the triangle decomposition step. If we visit a face vertex during the boundary traversal, we must move on directly to the other face vertex and then continue on from there.

Since the sweep-surface does not have to be planar, the cut-boundary need not lie in a plane, and thus the cut-boundary alone does not give sufficient information on how to triangulate the new surface. Obviously, this surface should lie on the sweep-surface. For general sweep-surfaces, the new surfaces can be triangulated using constrained 2.5-D Delaunay triangulation techniques [6] with the given cut-boundary and sweep-surface. In our implementation, we make use of the fact that the cut-boundary lies completely on the sweep-surface and that the sweep-surface can be parameterized in 2-D. Therefore, the cut-boundary is mapped into 2-D, where it can be triangulated more easily and efficiently, as can be seen in Figure 6. The new vertices created in the triangulation process are then finally mapped back into 3-D using the sweep-surface and the vertices' parameters.

For a realistic visualization, the new surfaces must also be textured. Because the triangulation process was conducted in 2-D, the new vertices' 2D-coordinates can be used directly as texture coordinates. As long as the distortions arising from the 2D-parameterization of the sweep-surface are small, this simple approach has shown to produce adequate results.

Since we now again have one (closed) surface mesh, performing another cut through both a new and an old surface mesh simultaneously is straightforward and works exactly the same way as described above.

6 RESULTS

We are able to perform cuts in real-time on models with 1000 tetrahedra and 2000 surface mesh triangles on a 3GHz Pentium IV machine. We can handle complete as well as partial cuts in both the volumetric and the embedded surface mesh (Figure 7).

(a) Polyp model.

	original mesh	traditional subdivision	hybrid (standard triangul.)	hybrid (enhanced triangul.)
tets	136	543	282	266
edges	265	813	497	477
nodes	69	201	123	115
tet_{minAR}	0.017	10^{-6}	0.015	0.017
V_{max}/V_{min}	25.07	6674.1	259.3	92.87
l_{max}/l_{min}	4.11	35.84	23.92	11.96
h_{max}	0.005	6×10^{-5}	5.7×10^{-4}	0.001

(b) Myoma model.

	original mesh	traditional subdivision	hybrid (standard triangul.)	hybrid (enhanced triangul.)
tets	277	1772	615	599
edges	421	2254	994	973
nodes	85	492	224	200
tet_{minAR}	0.025	10^{-7}	0.014	0.016
V_{max}/V_{min}	11.67	3.6×10^{7}	94.84	52.38
l_{max}/l_{min}	2.03	540.01	9.743	7.70
h_{max}	0.004	6×10^{-6}	0.0009	0.0012

Table 1: Comparison between our hybrid cutting approach and a standard subdivision approach as described in [3] where no existing nodes are separated. The element count for hybrid cutting is 2-3 times smaller, while mesh quality is kept acceptable. Using our newly proposed triangulation of tetrahedral faces (section 4.4) results in even better mesh quality, compared to other symmetric triangulation approaches.

We compare our hybrid method to existing approaches in Figure 9. First, a tetrahedral mesh is cut along existing faces, resulting only in a coarse approximation of the sweep-surface (Figure 9(a)). In comparison, decomposing all cut tetrahedra matches the sweep-surface exactly, but badly-shaped elements are created and the element count has drastically increased (Figure 9(b)). In Figure 9(c), the same mesh is cut using our hybrid approach without node-snapping. The sweep-surface is more closely approximated, while no badly-shaped elements have been created and the element count has increased only slightly. Finally, when node snapping is applied together with our hybrid approach as shown in Figure 9(d), the sweep-surface is approximated even better, without further increasing the element count.

In Table 1, our hybrid approach and the pure subdivision method according to [3] are compared. Element count as well as several mesh quality measures are considered. For our hybrid cutting, element count is 2-3 times smaller. The ratios between largest and smallest tetrahedron volume and the largest and smallest edge length are still satisfactorily small after a cut. Also, the lowest aspect ratio of all tetrahedra (tet_{minAR}; incircle radius divided by circumcircle radius; 0.33/0.00 being the highest/lowest possible values) does not become too small. For pure subdivision, aspect and volume ratios can become arbitrarily bad. All three measures are crucial for a stable deformation simulation. In addition, it can be seen from the larger aspect and smaller volume ratios that mesh quality is improved when using our newly proposed tetrahedral face

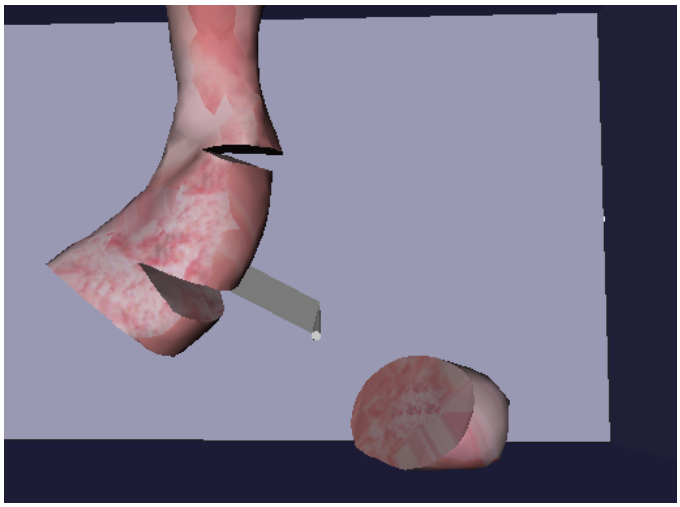


Figure 7: Several incisions have been made into a polyp.

triangulation (section 4.4), allowing us to use a simulation time step which is almost twice as large as for the standard approach. In our measurements, the maximum distance d of cut-nodes to the sweep-surface is one-fifth of the average edge length of the mesh. The models we have used are the polyp shown in Figure 9 and the myoma depicted in Figure 10.

Decoupling the simulation and visualization domains allows us to simulate highly detailed surfaces in real-time, using a low-resolution volumetric mesh. Creating a new surface after a cut is totally independent of the tetrahedral mesh, enabling us to create arbitrarily detailed surfaces and sharp but non-jagged edges. In Figure 8, we have cut an artificial deformable solid several times. Note the intricate surface detail such as sharp edges and very thin object parts, and the small portions of material we have cut off. Because our approach does not create small or badly-shaped tetrahedra, it is easy to perform several (possibly intersecting) cuts in the same region of the model.

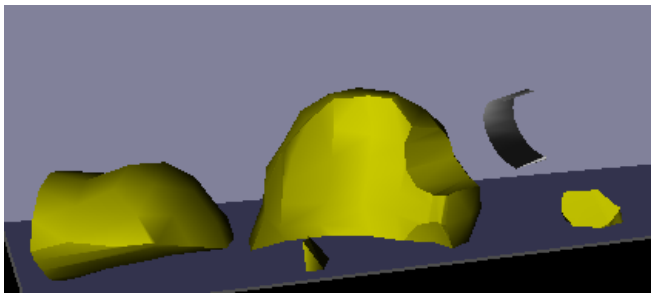
6.1 Integration into Surgery Simulator Environment

Our methods have been integrated into our hysteroscopy simulator. In this system, the methods have been combined with physical soft-tissue simulation, collision handling of deformable models, fluid simulation to simulate bleeding, as well as haptic interaction to build an extremely realistic environment for virtual reality based training of surgical interventions.

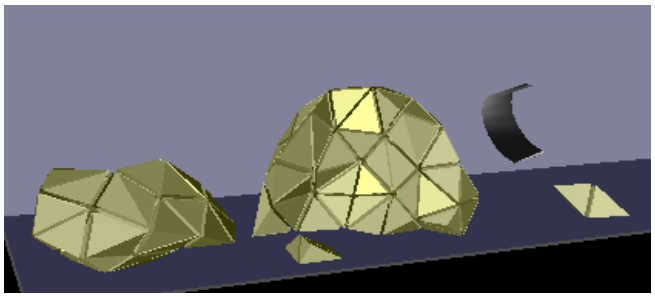
Figure 10 shows an example of a surgical procedure. In the upper image panel, the real-world situation can be seen. The surgeon uses the cutting tool to move and deform the myoma. She then moves the loop electrode behind the myoma, turns on the electrical current and pulls the electrode towards the front, thereby cutting off a portion of tissue. The process is repeated until everything has been ablated. In the bottom image panel, the same cutting process is shown in the virtual environment.

7 CONCLUSIONS AND FUTURE WORK

We have presented a new real-time approach for cutting tetrahedral meshes. It combines existing techniques of decomposing tetrahedra into smaller elements and cutting entirely along existing edges and faces. We are able to approximate a given sweep surface more closely while avoiding the creation of degenerate tetrahedra, thus keeping the physical simulation stable. Mesh relaxation improves



(a) The solid was cut into several pieces by a curved knife. Note the sharp and thin edges and the small portions we have carved out.



(b) The underlying tetrahedral mesh. The small pieces are represented by a larger, better-shaped tetrahedra.

Figure 8: Several scoops have been taken from an artificial deformable solid.

the quality of a cut mesh additionally. Decoupling the simulation and visualization domains allows the representation of intricate surface detail without placing an additional computational burden on the physical simulation.

Nevertheless, some simplifications were made, which we will address in the future. The cutting is not progressive, and there is no force feedback during the cutting. Moreover, we only model the result of a cut after it has been performed. Finally, our tetrahedral decomposition schemes are not optimal with respect to element count.

ACKNOWLEDGMENT

The authors would like to thank all developers of the hysteroscopy simulator project. This research has been supported by the NCCR Co-Me of the Swiss National Science Foundation.

REFERENCES

[1] D. Bielser, P. Glardon, M. Teschner, and M. Gross. A state machine for real-time cutting of tetrahedral meshes. In *Proc. of Pacific Graphics*, pages 377–386, 2003.

[2] D. Bielser and M.H. Gross. Interactive simulation of surgical cut procedures. In *Proceedings of the Pacific Graphics 2000*, pages 116–125, 2000.

[3] D. Bielser, V.A. Maiwald, and M.H. Gross. Interactive cuts through 3-dimensional soft tissue. In *Proceedings of the Eurographics '99*, volume 18, pages C31–C38, 1999.

[4] M. Bro-Nielsen. Finite element modeling in medical VR. *Journal of the IEEE*, 86(3):490–503, 1998.

[5] C.D. Bruyns and K. Montgomery. Generalized interactions using virtual tools within the spring framework: Cutting. In *Proceedings of Medicine Meets Virtual Reality 02/10*, pages 79–85, 2002.

[6] L.P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 274–280, New York, NY, USA, 1993. ACM Press.

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

[8] F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. In *Proceedings of the Eurographics 2000*, volume 19, pages C271–C282, 2000.

[9] F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno. Enabling cuts on multiresolution representation. In *The Visual Computer*, pages 274–286, 2001.

[10] F. Ganovelli and C. O'Sullivan. Animating cuts with on-the-fly remeshing. In *Eurographics 2001 - short presentations*, pages 243–247, 2001.

[11] A. Liu, F. Tendick, K. Cleary, and C.R. C.R. Kaufmann. A survey of surgical simulation: Applications, technology, and education. *Presence: Teleoperators & Virtual Environments*, 12(6):599–614, 2003.

[12] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph.*, 23(3):385–392, 2004.

[13] A.R. Mor and T. Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *CVRMed-Proceedings*, volume 19, pages 598–607, 2000.

[14] M. Mueller and M. Gross. Interactive virtual materials. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

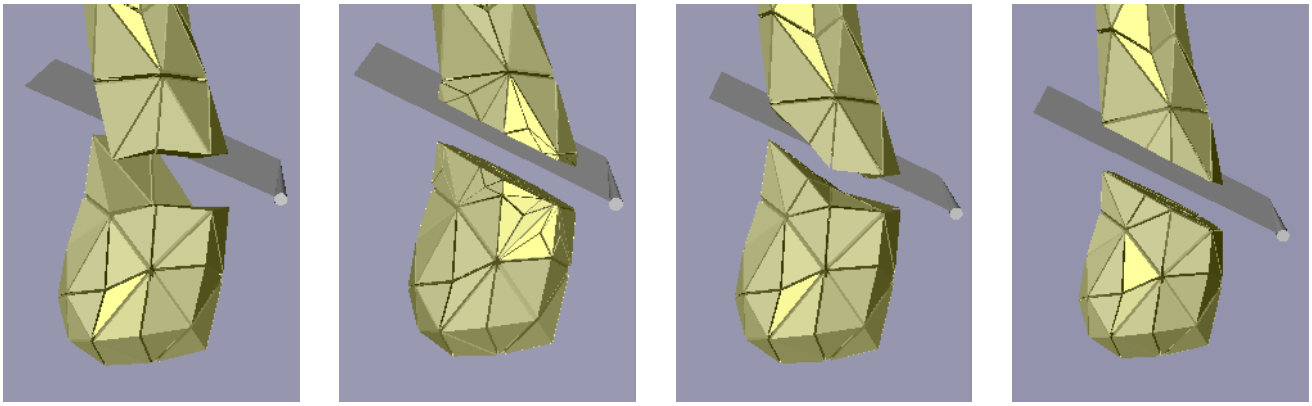
[15] H.-W. Nienhuys and A.F. van der Stappen. Combining finite element deformation with cutting for surgery simulations. In *Proceedings of the Eurographics '00*, pages 274–277, 2000.

[16] H.-W. Nienhuys and A.F. van der Stappen. Supporting cuts and finite element deformation in interactive surgery simulation. In *Tech. report UU-CS-2001-16, Univ. Utrecht*, 2002.

[17] J.F. O'Brien and J.K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146, 1999.

[18] D. Serby, M. Harders, and G. Szekely. A new approach to cutting into finite element models. In *Proceedings of the Fourth International Conference on Medical Image*, pages 425–433, 2001.

[19] M. Teschner, B. Heidelberger, M. Muller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 312–319, Washington, DC, USA, 2004. IEEE Computer Society.



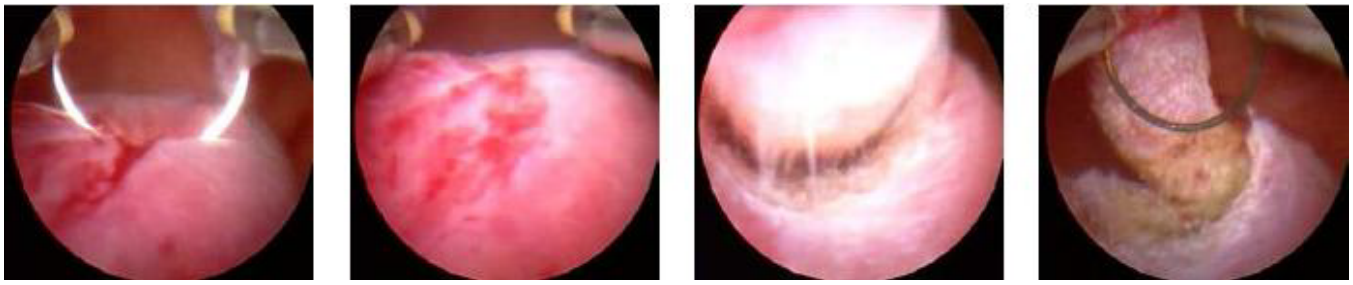
(a) A tetrahedral mesh is cut along existing edges, nodes and faces.

(b) The mesh is cut by decomposing each tetrahedron intersected by the sweep-surface.

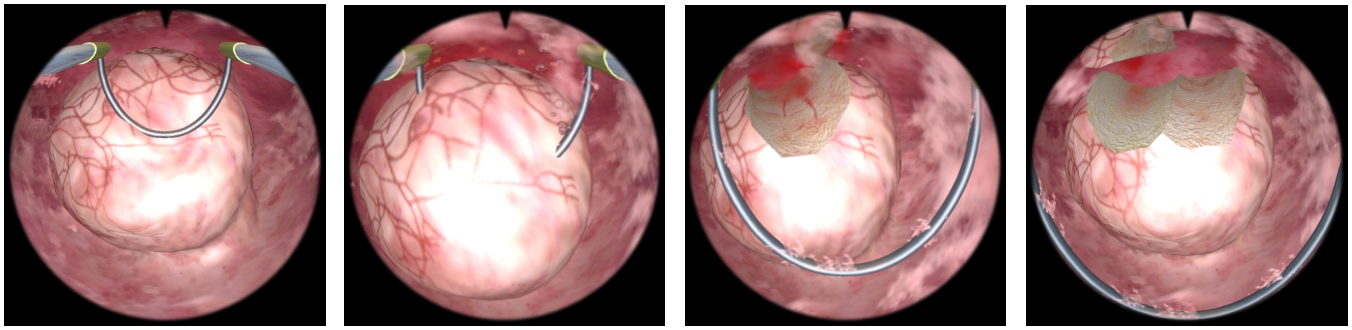
(c) The mesh is cut using our hybrid approach without node-snapping.

(d) After cutting the mesh with the hybrid approach, nodes are additionally snapped to the sweep-surface.

Figure 9: Comparison of our hybrid cutting approach with existing methods of cutting only along existing elements and splitting intersected tetrahedra.



(a) Cutting in a real operation.



(b) Cutting in the hysteroscopy simulator.

Figure 10: Comparison between cutting in a real operation (a) and cutting in the simulator environment (b). The surgeon uses the cutting tool to move and deform the myoma. She then moves the loop electrode behind the myoma, turns on the electrical current and pulls the electrode towards the front, hereby cutting off a portion of tissue. The process is repeated until everything has been ablated.