Modifying Soft Tissue Models: Progressive Cutting with Minimal New Element Creation

Andrew B. Mor and Takeo Kanade
Center for Medical Robotics and Computer Assisted Surgery
Carnegie Mellon University, Pittsburgh, PA 15213
{abm,tk}@ri.cmu.edu
http://www.mrcas.ri.cmu.edu

Abstract. Surgical simulation is a promising technology for training medical students and planning procedures. One major requirement for these simulation systems is a method to generate realistic cuts through soft tissue models. This paper describes methods for cutting through tetrahedral models of soft tissue. The cutting surface follows the free form path of the user's motion, and generates a minimal set of new elements to replace intersected tetrahedra. Intersected elements are progressively cut to minimize the lag between the user's motion and model modification. A linear finite element model is used to model deformation of the soft tissue. These cutting techniques coupled with a physically based deformation model increases the accuracy and applicability of a surgical simulation system.

1. Introduction

Surgical simulation is getting increased attention due to its promise of increasingly realistic modeling. It will be used by medical students when learning new procedures and will give surgeons the ability to simulate patient specific data when planning complex procedures. Two important parts of any surgical simulation system are soft tissue modeling and the ability to modify the simulated anatomical structures. Modifying the simulated tissue, as a cut is taking place, in a physically realistic and consistent fashion is an important part of any surgical simulation. Physically accurate real-time models are required to give the user a correct feel of how the soft tissue responds.

Previous work has been done on physically based tissue modeling and on techniques required to realistically modify an object's topology. But no work had addressed both topics simultaneously. Because the modification of the tissue model directly impacts the accuracy and speed of the soft tissue simulation, it is important to address both issues simultaneously. There has been no work addressing the problem of how to modify the topology of individual elements while a cut is taking place, an important issue if the nominal element size is easily discernible.

1.1 Prior Work

Soft tissue simulation can be implemented utilizing either surface or volumetric models. Volumetric models can simulate objects with interior structure that surface models can not encode. [5] gives a good overview of different methods used to simulate deformable objects. Finite element methods are based on a continuum model

S.L. Delp, A. M. DiGioia, and B. Jaramaz (Eds.): MICCAI 2000, LNCS 1935, pp. 598-608, 2000

and generate physically based results. [9] introduced models based on elasticity theory to the computer graphics community. [3] modeled human muscle using an FEM mesh. [2] developed a system utilizing classical, three dimensional solid finite element models using condensation, precalculation of the inversion and exploitation of the sparse structure of the force vector to accelerate the calculation of static deformations. [4] demonstrated a representation they named tensor mass based on linearly elastic continuum mechanics that are solved in a dynamic fashion over time. [11] recently introduced a non-linear finite element method that handles large-scale deformations better than linear methods.

There has been less attention applied to the problem of modifying deformable objects. Modification precludes any precomputation for the soft-tissue simulation. Cutting of surface-based finite element models was demonstrated by [8]. [6] created a static cutting surface by specifying the beginning and end points of a cutting edge, and then interpolating between them. Intersected elements were split according to the number of intersected edges. [1] made free-form cuts through an object by tracking the tip and the direction of a cutting edge. A small cutting plane was generated at every time stop between the previous edge position and the current edge position, and any element that is intersected is split into 17 smaller elements once the cutting tool leaves the element.

This paper is organized in the following manner: Section 2 states the proposed methods and contains an overview of the system, Section 3 describes minimal set cutting, Section 4 details the method for progressive cutting, Section 5 describes the soft tissue model, and Section 6 describes our results. The results and future work are then discussed.

2. Soft Tissue Simulation and Object Modification

To provide a convincing visual and haptic experience for surgical simulation, the simulation of the soft tissue must be performed in as realistic and accurate a manner as possible. Finite elements models generate the most accurate results, and are becoming more acceptable for simulation use as computing power increases. We use a linear finite element utilizing lumped parameters. This model is quickly and easily updated when changes to the topology of the model are made.

Cutting tissue is one of the main activities performed during surgery, and the results of such modification in a simulator can greatly affect both the sense of realism of the cutting process and the computational load on the host computer. Three items can greatly affect the cutting process. First, the simulation should allow cuts along free-form surfaces that the user traces out with a cutting tool. If the user is constrained to cut along regular grid lines, or only along boundaries between elements, the sense of realism will be greatly compromised. Cuts will not appear where the user expects them to be, and the resulting surface will either be artificially smooth or stair-cased in shape. Second, cuts should occur as the user moves the cutting tool through the object. If the cut lags behind, a disconnect will form between the user and the simulation. Third, if the cutting process generates a large number of smaller elements, the computational cost of the new elements may be so great so as to adversely affect the update rate of the

simulation. To address these issues, we propose the use of a method that generates a minimal set of new elements that follows the motion that the user traces out. We also propose generating a progressive, temporary cut, as the user moves the cutting tool within an element. Once the cut leaves an element, it is cut permanently.

2.1 System Overview

The soft tissue simulation system includes an input device, finite element modeler, and object modification routines. The general flow of the system, after start-up when the model is loaded and the individual element parameters are first determined, begins with gathering the position of the tool that the user is holding. The current implementation uses keyboard input for simplicity, but the system is easily extendable to using a 6DOF haptic device. Next, we determine the intersection state between the tool and the user. If any elements are partially intersected, where the cutting edge is still within the interior of the element, then temporary elements are created using the progressive cutting method. If any elements are fully intersected, the element is fully split into its minimal subset. Lastly, the internal elastic forces are determined, and then nodal positions are updated. The scene is then rendered to the user.

3. Minimal Set Cutting

The described cutting method creates a minimal set of new tetrahedra that contain the cut surfaces that follow the user's motions. There are two main activities required for cutting tetrahedral models: the detection and storage of intersections, both of edges and faces; and the subsequent subdivision of the intersected element when a cut is completed. First, intersections between the cutting tool and the model are detected and stored. Once an element is completely cut through, it is then subdivided based on the number and type of intersections among its edge and faces.

3.1 Intersection Detection

Cutting with a scalpel can be viewed as the motion of a finite length cutting edge passing through an object. If the body of the blade is ignored, and the edge is taken to be infinitely sharp, then the problem is reduced to tracking the passage of a line segment moving in time and space through a tetrahedral mesh [1]. Figure 1 shows the path of a cutting edge from time t_i to t_{i+1}, as it creates two face intersections and one edge intersection.

The swept surface created by the path of the cutting edge must be tested at every time step for intersections with the model. Two tests are required: the intersection of the path of the tip of the cutting tool and the faces of the tetrahedron, and the intersection of the swept surface and the edges of the tetrahedron. These two tests generate, respectively, face intersections, which mark the base of the cut, and edge intersections, which occur where the model is split in two by the cutting edge.

The procedure for updating the intersection state of the model starts with a global search to determine if the path of the blade has intersected the model. A global search

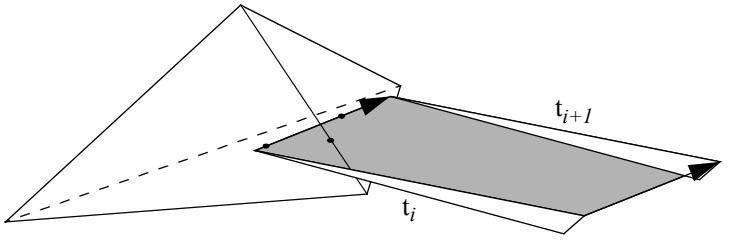

Figure 1. Cutting edge intersection with a tetrahedron.

is inefficient, but guarantees that all intersections will be detected. Other initial collision detection techniques can also be used. If any tetrahedron has been intersected, then all 6 of the tetrahedron's edges and all 4 of its faces are tested against the swept surface and cutting tip path. The current, deformed, positions of the vertices are used when testing for intersections. When an intersection occurs, the barycentric coordinates of the intersection are determined. The corresponding rest position of the intersection is then used when updating the soft tissue parameters of the new elements. After all the tetrahedra are tested, if any were intersected, the model is marked as having an intersection present.

Next, if the model has been intersected, all of the intersected elements are checked to see if the cutting instrument has either passed through any non-intersected faces or edges, or has left the tetrahedron. If a new intersection occurs, then the intersection information for that tetrahedron is updated. Neighboring elements that also contain the newly intersected edge or face are updated, thereby using spatial coherency to propagate the cutting motion through the model. If the cutting edge no longer passes through an intersected element, then the user has completed the cut, and the element will be subdivided.

3.2 Element Subdivision

Tetrahedral elements cut by planar, or near-planar, surfaces will fall into one of five different topological cases, based on the number of cut edges and intersected faces. There are two different cases where the tetrahedron is completely cut through into two pieces, and three cases where the element is cut, but not completely through. Figure 2 shows the different cases.

Within the framework of the tetrahedral mesh numbering, there are multiple orientations of the cut element based on the ordering of the vertices and cut edges. When the element is completely cut into two, there are four different permutations when three edges are cut, as in case *i*, and three different permutations in case *ii*, when four edges are cut. The four different permutations of case *i* correspond to each of the four vertices being cut away from the remaining three. When only one edge is cut, case *iii*, there are six permutations, and there are twelve permutations for both cases *iv* and *v*.

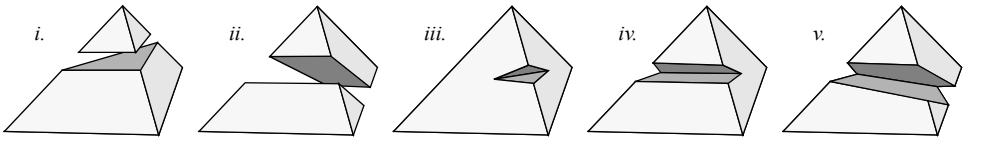

Figure 2. The five cases of tetrahedron subdivision after a completed cut.

Individual procedures were implemented for each type of intersection, so that no excess elements would be created. Only five to nine new elements are created for each cut element, depending on the type of intersection. This minimal subdivision uses only the original vertices of the element and vertices created due to the cutting action: one vertex at the point of each face intersection, and two vertices at the point of each edge intersection. Figure 3 demonstrates how each half of the cut element from case *ii* in Figure 2 is minimally subdivided. In this case, six elements are created to replace the original one.

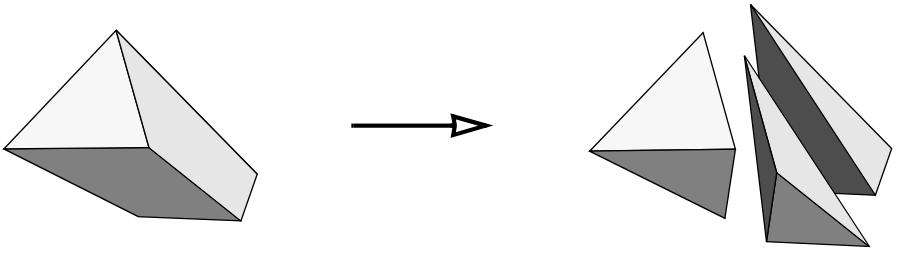

Figure 3. Minimal element subdivision.

Each procedure uses a lookup table to determine how to mirror or rotate the vertices to fit the default orientation. After the ordering is determined, any new edges that are needed are created. Then the new tetrahedra are created and the original tetrahedron is removed.

4. Progressive Cutting

Previous methods of modifying objects, and the basic technique described above, do not split an element until the cut has been completed. With current models, where the average element size can be quite large, this introduces a noticeable lag into the cutting process. We have implemented a method of progressive cutting that generates a minimal subdivision of a partially cut tetrahedron. The subdivision is always based on the geometry of the original element, thereby minimizing a potential source of error.

The general procedure for progressive cutting utilizes a temporary subdivision of each partially cut element. An example cut is shown in Figure 4. First, any temporary face intersections caused by the cutting edge are updated for each partially cut tetrahedron. A temporary face intersection occurs when the cutting edge intersects a face. This type of intersection does not occur for the permanent intersections described previously.Then, the modified topology of the partially cut element is checked for any changes. A change occurs when a new intersection is created: for example, when the element is first cut into, or when the cutting edge or tip passes through another edge or face. If the topology has changed, a new minimal set of temporary tetrahedra are created and all the old temporary tetrahedra are removed. If the modified topology has not been changed, then the temporary elements are updated using the new positions of any temporary face intersections. Once the cutting edge leaves an element and the cut is completed, the temporary elements are removed, and a final subdivision is created.

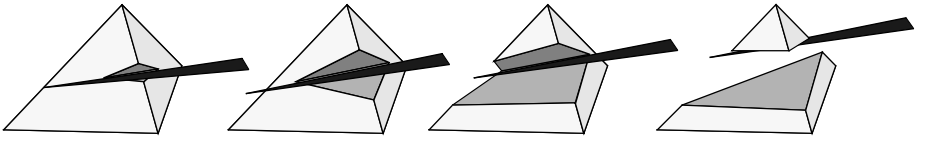

Figure 4. Progressive cutting example.

There are eleven different combinations of intersected edges, faces, and temporary face intersections, which are enumerated in Table 1. Momentarily marking temporary intersections as permanent, many of the cases can directly use the procedures described in the previous section on minimal cutting. The cases which are unique to progressive cutting were implemented in a similar fashion as described above, with a minimum number of new elements created for each cut.

Table 1. Enumeration of different cases for temporary intersections.

Case:	1	2	3	4	5	6	7	8	9[a]	10	11
Edge Ints.	0	0	1	1	1	2	2	2	2	3	3
Face Ints.	1	2	0	1	2	0	1	2	2	0	1
Temp. Face Ints.	1	2	2	1	2	2	1	2	2	2	1
Interior Ints.	1	0	0	1	0	0	1	0	0	0	1

a. Same number of intersections as case 8, but case 9 has a different subset of edges intersected.

There are two important issues unique to progressive cutting that must be addressed: how to deal with the tip of the cutting edge being within the interior of an element; and

how to deal with two intersections on one face, which happens when a face is intersected permanently by the passage of the tip of the blade and temporarily by the cutting edge.

When the tip is within the interior of an element, ideally we would want the model to be able to open up along the cut of the blade, so that the user could see all the way up to the base of the cut, where the tip is located. But given the nature of the subdivision for a generic cut, this would not be possible. An example of this is shown in Figure 5, case *i*, which corresponds to Case 1 from Table 1. In this case, the tip of the blade is inserted fully through one face, with the tip of the cutting edge within the interior of the tetrahedron. I_i is the tip of the cutting edge, I_f is a permanent face intersection, and I_{tf} is a temporary face intersection. There are now two intersections on one face. Ideally, as described above, the object would be able to open up along the edges between the two face intersections. But if no intermediate nodes are inserted between the two face intersections, a straight line will always connect them, and the model will not be able to open up. The fact that the model can not open up along these edges allows us to ignore the location of the tip within the model, and, in fact, to generate an arbitrary topology within the interior of the original element, since that topology will never be seen.

When there are two intersections on one face, we may ignore the fact that the model should open up, but we still have to make sure that none of the triangles generated on that original face overlap. This would occur, as shown in Figure 5, as the blade travels from case *ii* to case *iii*, when a temporary face intersection moves across an edge belonging to the other face intersection. The shaded area shows the overlapping area of the two triangles. If this occurs, then the modified topology of the partially cut tetrahedron has changed, and a new set of tetrahedra will be created, as shown in case *iv*.

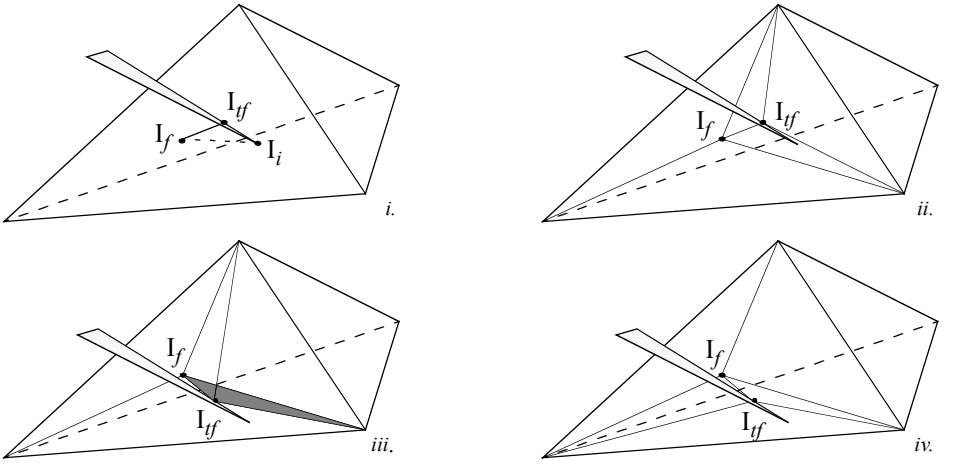

Figure 5. Temporary subdivision, with two intersections on one face.

5. Soft Tissue Modeling and Linear Finite Elements

The problem of modeling soft tissue can be approached in many different ways, depending on the trade-off of speed vs. accuracy. We implemented a linear finite element model due to its increased realism when compared to mass spring or surface based models. The mass and damping terms are diagonalized and lumped at the nodal positions. The system is explicitly solved to integrate the Newtonian dynamics of the nodal masses.

The linear finite element model was generated in a manner similar to [4]. The global stiffness matrix is set up, then individual nodal and edge terms are broken out of the global matrix, and stored with the individual nodes and edges. The stiffness matrix is calculated in the standard manner [12].

Given an isoptropic solid, the equation for the element stiffness matrix, $\boldsymbol{K}^e$, simplifies to:

$$\boldsymbol{K}^e = V^e \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \quad . \tag{1}$$

where the matrix $\boldsymbol{D}$ is determined by material properties and the matrix $\boldsymbol{B}$ encodes the shape functions of the element**.** For a tetrahedron, this 12x12 stiffness matrix can be viewed as being composed of four 3x3 nodal displacement stiffness matrices, $\boldsymbol{K}_{ii}^e$, and six edge displacement stiffness matrices, $\boldsymbol{K}_{ij}^e$. The first stiffness matrix relates to the force felt by a node due to its own displacement from its home position, while the second matrix determines the force felt by the node *i* due to the displacement of node *j* from node *j*'s home position; this can be viewed as an edge effect since it occurs between two nodes.

To calculate the internal elastic force acting on a node *i*, the contributions from all the tetrahedra that node *i* belongs to are summed:

$$\boldsymbol{f}_i = \boldsymbol{K}_{ii} \overrightarrow{P_i^o P_i} + \sum_{j \in N(P_i)} \boldsymbol{K}_{ij} \overrightarrow{P_j^o P_j} \quad . \tag{2}$$

where $\boldsymbol{f}_i$ is the nodal force, $\boldsymbol{K}_{ii}$ is the sum of the $\boldsymbol{K}_{ii}^e$ tensors associated with all the tetrahedra that node *i* belongs to, P_i^o is the home position of node *i*, the tensor $\boldsymbol{K}_{ij}$ is the sum of the $\boldsymbol{K}_{ij}^e$ tensors associated with the edge from node *i* to node *j*, and $N(P_i)$ is a list of all of the nodal neighbors of the node *i*.

6. Results

Figure 6 shows the results of cutting a section off the left lobe of a liver model. Figure 7 shows the results of cutting through a rectangular object that is under tension. There were 576 tetrahedra in the rectangular object before cutting, on a 4x6x4 cubical lattice, and 954 afterwards. 60 elements were cut, removed, and replaced by a new set of 312 tetrahedra, an average of 5.2 elements added for every element removed. This

compares very favorably to a similar cut demonstrated in [1], where 110 elements were cut and 1870 new elements were created.

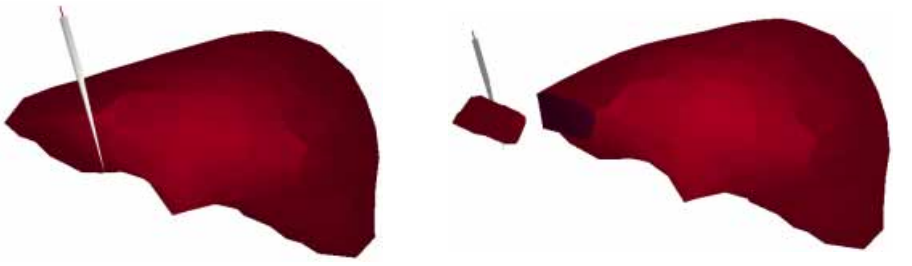

Figure 6. Cutting off a section of the liver. (Model courtesy of Project Epidaure, INRIA.)

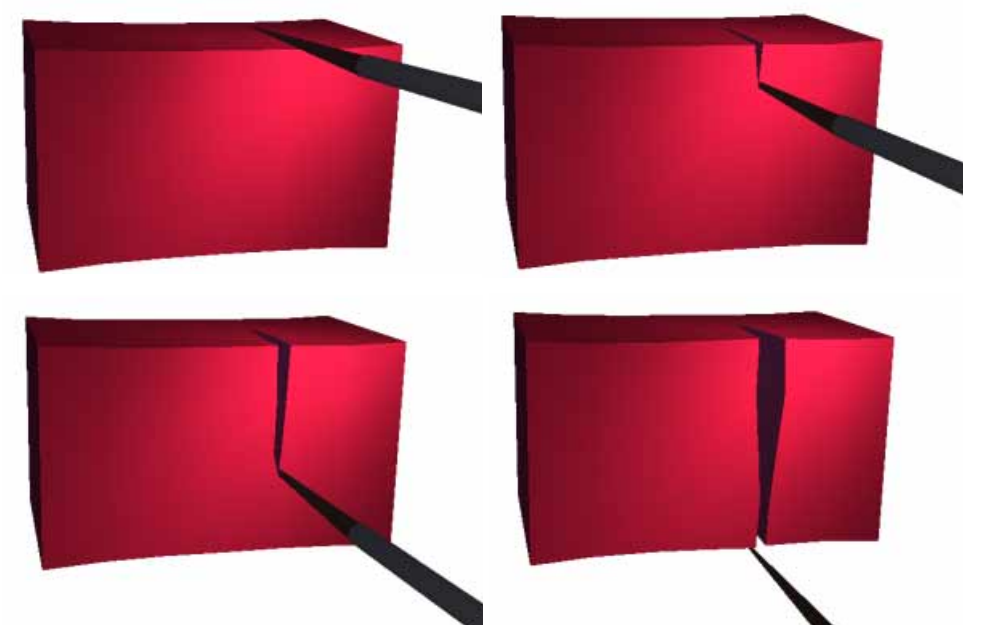

Figure 7. Cutting of a rectangular object under tension.

There is one major limitation with the current implementation. When the user first starts cutting into an element, or when a final cut creates an element that is very small, the model can become unstable. This occurs because the stiffness matrices for the small elements are quite large, and even small displacements can generate very large

forces. This is not a problem with the cutting techniques, only with the soft tissue modeling. Because of this instability, the examples demonstrated were performed on a graphical simulator, without haptics. The examples were also run with a very small time step, to insure the stability of the soft tissue model, and were not real-time.

7. Discussion

We have presented a technique for realistically modifying the topology of simulated objects in an efficient manner. The technique of minimal cutting generates a minimum set of new elements, and therefore is useful in a wide range of applications where the computational load and update rates are factors in the realism of the simulation. Progressive cutting is useful in situations where the size of individual elements is noticeable to the user and any apparent lag between the motion of the cutting tool and the modification of the model would be distracting. The described techniques work on any tetrahedral mesh, irrespective of the overall shape of the model. The soft tissue simulation will run at update rates around 100Hz on the liver model, given the size of the model and results demonstrated in [4]. This will be sufficient to interpolate for output through a force feedback device.

Future work will be directed towards fixing the limitations imposed by the use of the linear finite element model. Linear models are only valid up to approximately 10% deformation, so a non-linear model will be investigated. Also, the problems caused by small tetrahedra will be addressed. One technique currently under investigation is to check the length of all new edges and the height of the new tetrahedron with respect to a minimum edge length. If the new element does not pass the check, then snap the position of the new vertex to the nearest original edge.

Acknowledgments

We would like to thank the Project Epidaure Team at INRIA for the model of the liver.

References

[1] 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.

[2] M. Bro-Nielsen and S. Cotin, “Real-time Volumetric Deformable Models for Surgery Simulation Using Finite Elements and Condensation”, Proceedings of Eurographics ‘96.

[3] D. T. Chen and D. Zeltzer. “Pump It Up: Computer Animation of a Biomechanically Based Model of the Muscle Using the Finite Element Method.” Computer Graphics (SIGGRAPH), num 26, July 1992.

[4] Cotin, S., Delingette, H., Ayache, N., “Efficient Linear Elastic Models of Soft Tissues for Real-Time Surgery Simulation”, INRIA T.R. No. 3510, October 1998.

[5] S. Gibson and B. Mirtich. “A Survey of Deformable Models in Computer Graphics.” TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, 1997.

[6] Mazura, A., Seifert, S., “Virtual Cutting in Medical Data”, Medicine Meets Virtual Reality, San Diego, CA, 1997.

[7] Reznik and Laugier, “Dynamic Simulation and Virtual Control of a Deformable

Fingertip", Proceedings of IEEE International Conference on Robotics and Automation, Minneapolis, MN, 1996.

[8] Song, G., Reddy, N., "Towards Virtual Reality of Cutting: A Feasibility Study", Proceedings of 16th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Baltimore, MD, 1994.

[9] Terzopolous, D., Platt, J., et al. "Elastically Deformable Models", Computer Graphics Proceedings, Annual Conference Series, Proceedings of SIGGRAPH 87, pp 205-214, 1987.

[10] Terzopolous, D. and Waters, K. "Physically-Based Facial Modeling, Analysis, and animation", Journal of Visualization and Computer Animation, 1:73-80, 1990.

[11] Y. Zhuang and J. Canny. "Real-time Simulation of Physically Realistic Global Deformation." IEEE Vis'99. San Francisco, California. October 24-29, 1999.

[12] Zienkiewicz, O., Taylor, R., The Finite Element Method, McGraw-Hill Book Co., London, Fourth Edition, 1988.