A Surgery Simulation Supporting Cuts and Finite Element Deformation

Han-Wen Nienhuys and A. Frank van der Stappen

Institute of Information and Computing Sciences, Utrecht University
PO Box 80089, 3508 TB Utrecht, The Netherlands
{hanwen,frankst}@cs.uu.nl

Abstract. Interactive surgery simulations have conflicting requirements of speed and accuracy. In this paper we show how to combine a relatively accurate deformation model—the Finite Element (FE) method—and interactive cutting without requiring expensive matrix updates or precomputation. Our approach uses an iterative algorithm for an interactive linear FE deformation simulation. The iterative process requires no global precomputation, so runtime changes of the mesh, i.e. cuts, can be simulated efficiently. Cuts are performed along faces of the mesh; this prevents growth of the mesh. We present a provably correct method for changing the mesh topology, and a satisfactory heuristic for determining along which faces to perform cuts. Nodes within the mesh are relocated to align the mesh with a virtual scalpel. This prevents a jagged surface appearance, but also generates degeneracies, which are removed afterwards.

1 Introduction

In a surgery simulator, surgeons can train surgical procedures on virtual patients. Such simulators offer a promise of reducing costs and risks when training surgeons. A training simulation can only replace training on live patients if it is realistic enough, and it can only reduce costs if it does not require big resources. Hence, the challenge in virtual surgery is to produce higher realism with as little computing resources as possible.

A full-fledged interactive surgery simulator consists of a core system that computes tissue response to surgical manipulations. It is supported by a display system for visual output and a force-feedback device for haptic output. The underlying problems in the core system, i.e. the soft tissue simulation, form the focus of our research.

A soft tissue simulation must provide convincingly accurate visual and haptic response to manipulations such as poking, pulling, cutting and cautering. We have chosen to simulate the elastic manipulations using the Finite Element Method (FEM). This method is based on a physical model of deformation, and solves the constitutive equations induced by that model. This makes the FEM potentially accurate from a physical point of view.

W. Niessen and M. Viergever (Eds.): MICCAI 2001, LNCS 2208, pp. 145–152, 2001.

We think that a physically accurate model will ultimately be preferable to ad-hoc heurististic models, such mass-spring models. Therefore we have chosen to use the simplest model available, static linear elasticity with linear geometry. Despite inaccuracies, we think that building a surgery simulation using this idealized model is a first step on the way to a physically accurate surgery simulation.

It has been suggested before that the FEM offers high fidelity tissue simulation. Bro-Nielsen [3], who first tried to use the FEM for surgery simulation, concluded that FEM is incompatible with cutting operations, because updating precomputed matrix inverses is too expensive for on-line operation. In our approach, we avoid these costly updates by using an iterative method that does not require global precomputed structures. To our best knowledge, this makes us the first to combine efficient cutting with an interactive FEM simulation [10].

Other prior work in deformation mostly uses heuristic models, such as Chain-Mail [11] or mass-spring models. These models are relatively straightforward to understand and implement, but they lack a rigorous physical basis. Recent advances have tried to incorporate aspects of the FEM into mass-spring models, blurring the distinction between these two approaches of soft-tissue simulation. For example, the hepatic surgery simulator project at INRIA uses uses a mass-spring model where the forces are inspired by linear and non-linear FEM, yielding what they call a tensor-mass model [5].

In the absence of cutting, full FEM implementation using matrix factorization have shown to be feasible. For example James and Pai [8] have produced interactive linear FEM simulations by removing internal nodes of the mesh. This is an algebraic procedure, and is more or less equivalent to precomputing a matrix inverse. Zhuang and Canny [12] achieve interactive non-linear deformation, also by matrix precomputations.

We combine the FEM for soft tissue simulation with algorithms for cutting. Since the deformation operates on a mesh, the cutting problem boils down to cutting in meshes. The basic problem with cutting in meshes is that one tetrahedron sliced with a blade does not yield two tetrahedra. The same holds for any other finite class of polygonal solids.

To accomodate cuts, the system has to contain routines that adapt the mesh to make cuts appear where a user performed them. Of course, this routine can only use a restricted set of shapes, e.g. tetrahedra. We have attempted to come up with an approach that does not significantly increases the size of the mesh. This is a desirable property, since the performance of the deformation simulation is directly proportional to the mesh size.

Earlier work solved cutting operations simplistically by removing those parts of the mesh that came in contact with a surgical tool [4]. In more recent work involving cuts, the basis is some deformable model defined on a tetrahedral mesh, with a subdivision scheme to accomodate cuts. Bielser was the first to demonstrate fully volumetric cuts, albeit with an expensive scheme [2] that was later refined [1]. Other efforts include a dynamic level-of-detail model [6] and other

subdivision schemes [9]. A common characteristic of subdivision schemes is that they increase the mesh size, which degrades the performance of the simulation.

In this paper we discuss our FEM deformation scheme first. Then we discuss the cutting algorithm. We conclude with a discussion.

2 Finite Element Deformation

In the Finite Element Method (FEM), the material under scrutiny is subdivided in simple elements. If one assumes that these elements deform in a limited number of ways, then the behavior of the entire subdividision can be computed, yielding an equation that relates deformation and elastic force.

We have chosen a tetrahedral elements for the mesh. Such a mesh is the simplest mesh to allow arbitrary shapes to be constructed in 3D. In the linear elasticity model, the relation between displacements of the nodes in a tetrahedron τ and elastic force $\mathbf{f}_{v,\tau}$ exerted by that tetrahedron on a node v is linear. The total elastic force $\mathbf{f}_v$ on v then is the contribution from all tetrahedra incident with that node

$$\mathbf{f}_v = \sum_{\tau,\tau \ni v} \mathbf{f}_{v,\tau} \tag{1}$$

As can be seen, this quantity f_v depends only on displacements of nodes that are connected to v by an edge. Hence, we can calculate the elastic force locally for every node in the mesh.

The total deformation is determined by balancing all external forces with all elastic forces. In an equation, this is described as

$$f_{\text{external}^G} = -K^G u^G. \tag{2}$$

In this equation f_{external^G} and u^G represent the total external force on and displacement of the tissue respectively. Both are vectors of dimension $3n$, where n is the number of nodes in the mesh. The matrix K^G combines all force-displacement relations in one big $3n \times 3n$-matrix called *global stiffness matrix*. In fact, every entry of the vector $K^G u^G$ is an elastic force that can be computed by Equation (1); in other words: $K^G u^G$ can be computed using only local information, i.e. without generating the matrix K^G itself.

This insight is the key to our solution method: we use an iterative method that only requires calculating $K^G v^G$ for some $3n$-vector v^G in every iteration. In our implementation, we have chosen the conjugate gradient algorithm [7], but any other algorithm of the Krylov-subspace type could also be used.

We have found the conjugate gradient algorithm to be efficient enough for interactive use. For larger models, the solution does not appear instanteneous, though: the solution process is noticeable, and appears as a quickly damping vibration of the object. Our simulation is static, so the relaxation time and vibration frequencies have no obvious relation with temporal behavior in reality. It is therefore not clear how much this will detriment the realism of a real simulation.

On a single 500 Mhz Xeon CPU we can manipulate a model of 7986 tetrahedra and 1729 nodes interactively. The relaxation process runs at approximately 50 iterations per second. In this mesh, the Lamé-material parameters are $\lambda = \mu = 1$, which corresponds with Poisson ratio $\nu = 0.25$. For this material and this model, we observed that this model generally requires between 70 and 200 iterations to reach convergence.

3 Cuts

Surgery can only be meaningfully simulated if there is support for changing the tissue. We have taken the operation of cutting as a basic task to model. Cutting in our FE simulation amounts to cutting in a tetrahedral meshes. The problem with cutting in a tetrahedral mesh, is maintaining both tetrahedrality and making the cut appear where the user performed it. Most previous work proposed subdivision methods to achieve this: every tetrahedron that is encountered during a cut is subdivided to accomodate the scalpel path. There are two inherent problems with this approach: Firstly, the mesh will always grow more complex, which brings down performance. Secondly, If the scalpel passes close by features of the mesh, then unconditional subdivision leads to degeneracies, such as elongated or flattened tetrahedra. These degeneracies can cause numerical problems in the simulation.

In an attempt to overcome these problems, we have taken the extreme opposite of subdivision: our cutting method does not perform any subdivision at all. We achieve this by adapting the mesh locally so that there are always triangles on the scalpel path, and performing the cut along those triangles. This procedure can be subdivided into three subtasks, schematically shown in Figure 1.

- Selecting faces from mesh for dissecting and repositioning. We call this process *surface selection*.
- Repositioning nodes from the mesh so the selected faces are on the scalpel path. We call this process *node snapping*.
- Modifying the mesh to reflect the incision. We call this process *dissection*.

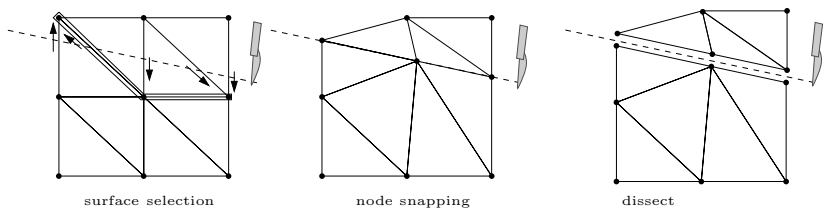

Fig. 1. Three steps in performing a cut, shown in 2D

Faces are selected on basis of edge-scalpel intersections: for each tetrahedron, we intersect the scalpel path with all edges of the tetrahedron. From every edge of the tetrahedron, we select the node closest to the intersection point. The set of nodes that is acquired in this way generally has three or less points, and represents a feature of the tetrahedron. The set of faces selected in this way is an approximation of the scalpel path: it is always close to the scalpel sweep, it is connected, and it generally does not branch.

Nodes are repositioned by projecting them orthogonally onto the scalpel path. For nodes that are on the boundary, this projection is done within the surface triangle containing the intersection. This minimizes the shape change caused by node repositioning.

Modifying the structure of the mesh is achieved by a flexible data structure. We have chosen a data-structure that closely mirrors a theoretical mesh description by means of simplicial complexes. This enabled us to construct verifiable algorithms to perform the modification.

Finally, the entire mesh has to be checked for connectivity: new loose components can be created by the cuts. To guarantee the existence of a solution, all new loose components must be fixed.

We have implemented this scheme, and succeeded in our goal: performing cuts without enlarging the mesh complexity. Figures 3 and 4 show the result for a single straight cut. The result of node snapping is clearly shown here. Figure 2 shows a large curved cut.

We also encountered a number of problems with this cutting scheme:

- A face is only selected when the scalpel crosses a tetrahedron completely. This means that there will be a lag between the cut instrumented by the user and the cut realized in the mesh.
- Node snapping cannot always be done without changing the external shape of the mesh: for instance, when the scalpel enters close to a corner of the mesh, that corner will move. In Figure 1, this happens with the upper left corner.
- Projecting nodes within the mesh can causes degeneracies: if four nodes of one tetrahedron are selected, node snapping will flatten that tetrahedron.

The last situation is relatively rare. It depends on the starting quality of the mesh, and the orientation of the scalpel path relative to the tetrahedralization. We found that the number of degeneracies were in the order of 5 % of the number of faces dissected for a typical cut in a Delauney tetrahedralization of a 8000 element cube.

However, when a degeneracy happens, it causes serious problems in the deformation routines, since a zero-volume tetrahedron respond to forces with infinite deformations. This prompted us to implement a routine that removes these degeneracies. After each cut, all tetrahedra of the mesh are scrutinized. Tetrahedra with low volumes or suspect aspect-ratios are classified according to their shape, and then an attempt is made to remove them. This search is repeated until there are no degeneracies left.

We observed that in most cases, the degeneracy removal is successful, and the number of tetrahedra only increases slightly due to the degeneracy removal, in the order of 1 %. Unfortunately, our current algorithm cannot repair all types of degeneracies.

4 Discussion

We have made a twofold contribution: first, we have described how to implement a static FEM analysis without requiring precomputation, thus enabling interactive cutting. We have obtained very satisfactory results, with deformations running at interactive rates for models with 2000 nodes.

Like all previous contributions in interactive surgery simulation, we assume a number of idealizations. We plan to refine our approach to deformation to obtain a more realistic simulation. At present, the simulation assumes the following:

- Force and deformation are proportional. This is the *linear material model*. Soft tissue has highly non-linear behavior: its resistance to stretching increases at higher stress levels.
- The deformation is assumed to be small. This is the *linear geometry approximation*. This assumption is valid when analyzing relatively stiff structures. Soft tissue is not stiff, which easily leads to big deformations.
- The simulation is static, i.e. only equilibrium situations are simulated, neglecting effects such as vibrations, inertia and viscosity.

Second, we have experimented with a cutting approach that does not increase the size of the mesh per se. We have had mixed success. On one hand, we learned that it was possible to execute this idea. On the other hand we found some serious drawbacks. There is an inherent lag between the scalpel and the realized cut. This effect may be alleviated by finely subdividing the mesh near the cut-area. However, this negates the original purpose of not increasing the mesh size. Second, we found that repositioning nodes within the mesh can easily generate degeneracies, which in turn have to be eliminated.

This experience prompts us to seek future work in cutting in a more flexible approach. In effect, other work unconditionally subdivides the mesh near the cut, while we unconditionally do not subdivide. In the future we will look at an approach that takes a middle road and retetrahedralizes the cut-area on demand, so that the mesh will be as fine as needed.

References

1. Daniel Bielser and Markus H. Gross. Interactive simulation of surgical cuts. In *Proc. Pacific Graphics 2000*, pages 116–125, Hong Kong, China, October 2000. IEEE Computer Society Press.
2. Daniel Bielser, Volker A. Maiwald, and Markus H. Gross. Interactive cuts through 3-dimensional soft tissue. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 31–38, 1999.

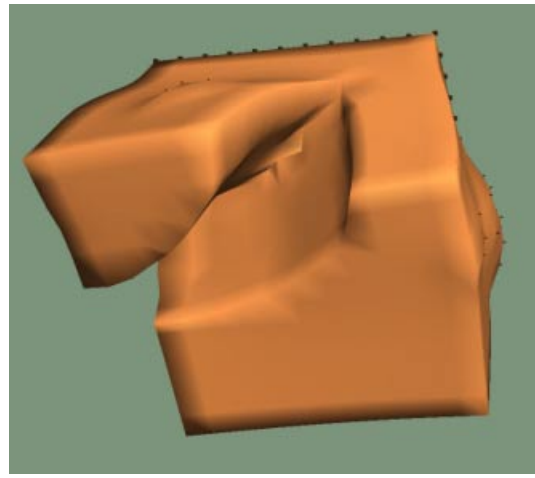

Fig. 2. A $12 \times 12 \times 12$ cube with a big cut. The material has Lamé-parameters $\lambda = \mu = 1$. The rear face of the cube is fixated.

3. Morten Bro-Nielsen. *Medical Image Registration and Surgery Simulation.* PhD thesis, Dept. Mathematical Modelling, Technical University of Denmark, 1997.
4. 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.
5. G. Picinbono and H. Delingette and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001: IEEE International Conference Robotics and Automation*, Seoul, Korea, May 2001.
6. Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. *Computer Graphics Forum*, 19(3):271–282, 2000.
7. Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* John Hopkins University Press, Baltimore, Maryland, 1983.
8. Doug L. James and Dinesh K. Pai. Artdefo—accurate real time deformable objects. In *SIGGRAPH*, pages 65–72, 1999.
9. Andrew B. Mor and Takeo Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, number 1935 in Lecture Notes in Computer Science, pages 598–607. Springer-Verlag, 2000.
10. Han-Wen Nienhuys and A. Frank van der Stappen. Combining finite element deformation with cutting for surgery simulations. In A. de Sousa and J.C. Torres, editors, *EuroGraphics Short Presentations*, pages 43–52, 2000.
11. Markus A. Schill, Sarah F. F. Gibson, H.-J. Bender, and R. Männer. Biomechanical simulation of the vitreous humor in the eye using an enhanced chainmail algorithm. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 679–687, 1998.
12. Yan Zhuang and John Canny. Real-time global deformations. In *Algorithmic and Computational Robotics*, pages 97–107, Natick MA, 2001. A. K. Peters.

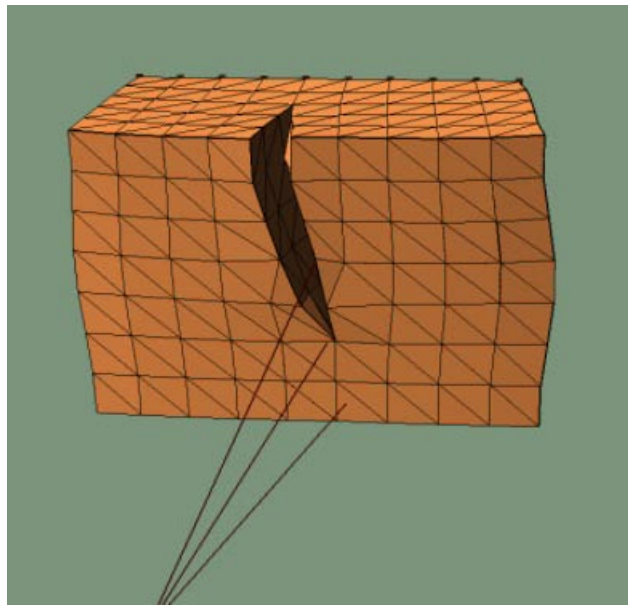

Fig. 3. A incision in a mesh with 480 nodes and 1890 tetrahedra, material parameters $\mu = \lambda = 1$. A dilating force is applied to the left and right. Edges are shown to demonstrate repositioning. The last scalpel movement and the next scalpel movement are indicated by the two partially penetrating triangles.

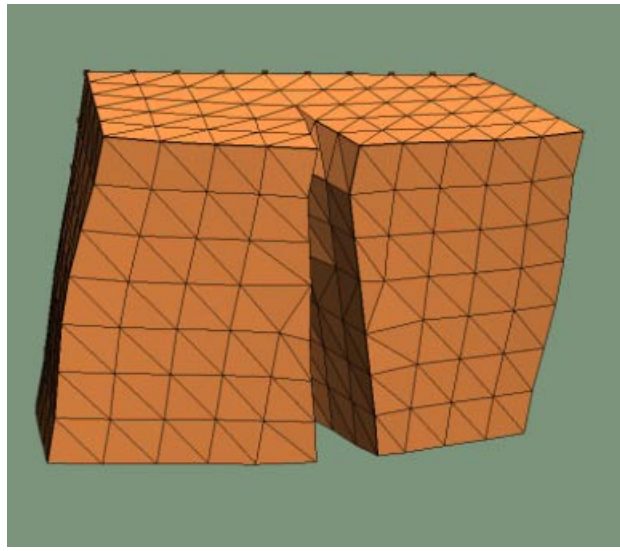

Fig. 4. The same incision, now completed.