Supporting cuts and finite element deformation in interactive surgery simulation

Han-Wen Nienhuys and
A. Frank van der Stappen

UU-CS-2001-16
June, 2001

Supporting cuts and finite element deformation in interactive surgery simulation

Han-Wen Nienhuys
A. Frank van der Stappen
Institute of Information and Computing Sciences
Utrecht University, 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 run-time changes of the mesh—that is, cuts— can be simulated efficiently. Cuts are performed along faces of the mesh; this prevents growth of the mesh without violating mass preservation laws. We present a robust 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. On the other hand it generates degeneracies, which are removed afterward.

1 Introduction

In an interactive surgery simulation, surgeons can train surgical procedures on virtual patients. Such simulations offer a promise of reducing costs and risks when training surgeons. If such a simulation is to replace training on living patients, then it must be realistic enough, and it can only reduce costs if it does not require many resources. Hence, the challenge in virtual surgery is to produce higher realism with as little computing resources as possible.

A full-fledged interactive surgery simulation 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 input and output. Problems in the core system, i.e., the soft tissue simulation, form the focus of our research.

A soft tissue simulation must provide convincingly realistic visual and haptic response to manipulations such as poking, pulling, cutting and cauterizing. We have chosen to simulate elastic response 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.

We think that a physically accurate model will ultimately be preferable to ad-hoc heuristic models, such as mass-spring models. As a starting point we have chosen to use the simplest FEM model available, static linear elasticity with linear geometry. Despite its 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.

In the static FEM for elasticity, the solution to a deformation problem is given by a configuration that minimizes the potential in the material. In static linear elasticity, the minimum energy configuration is given by a linear equation $f = -Ku$. Here f expresses external forces and u the deformation. The *stiffness matrix* K is a matrix that links both quantities. The sizes of vectors f and u are proportional to n, the number of nodes in the mesh.

It has been suggested before that the FEM offers high fidelity tissue simulation. Bro-Nielsen [1] first tried to use the FEM for surgery simulation. He used a number of off-line precomputations to speed up the on-line simulation. His technique essentially amounted to computing K^{-1} in advance, an operation that cost $\mathcal{O}(n^3)$ operations in his implementation. The inverse of the sparse matrix K is dense, but if f is sparse, then a guaranteed response time of $\mathcal{O}(n)$ is possible during the simulation. The price paid for this guarantee is a high start-up time. Bro-Nielsen also concluded that the FEM is incompatible with interactive cutting operations, because updating K^{-1} 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, mass-preserving cuts with an interactive FEM simulation [2].

Other prior work in deformation mostly uses heuristic models which are relatively straightforward to understand and implement, but lack a rigorous physical basis. One of these models is ChainMail [3, 4]. It uses a simplistic model of deformation. Nodes are arranged in structured ('voxel') meshes, and each pair of neighboring nodes can move within minimum and maximum distance constraints. Either a node moves freely, or a constraint is tight, and the node moves in concert with a neighbor. This mechanism provides a computationally cheap way to deform a model, but it is not realistic, since the virtual tissue does not offer an elastic response to force.

Most surgery simulations nowadays seem to be based on mass-spring models. Similar to FEM methods, the tissue is represented as a mesh. Springs are attached to edges of the mesh. In reaction to external force, the system of springs will strive for a minimal potential energy configuration. However, this minimal energy configuration is usually not found directly, but by means of a dynamic system. Mass-points are attached to the nodes. The mass/spring mesh forms a dynamic system subject to Newton's laws of mechanics. Numerical integration methods are used to compute the evolution of the system. Varying the characteristics of the springs allows various material properties to be simulated.

Mass spring models for interactive deformable models were first used by Terzopoulos to simulate facial expressions [5]. They have subsequently appeared in many simulations [6–8]. Recent advances have tried to incorporate aspects of the FEM into mass-spring models, blurring the distinction between these two approaches to soft-tissue simulation. For example, the hepatic surgery simulator project at INRIA uses a combination of precomputed linear static FEM and a

dynamic *tensor-mass* model—a mesh of mass points subject to forces derived from a nonlinear FEM formulation, instead of forces from springs [9, 10].

In the absence of cutting, full FEM implementations using matrix factorization have shown to be feasible. For example James and Pai [11] 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 an interactive dynamic nonlinear deformation, also by matrix precomputations. Székely et al. [13] employ a massively parallel hardware to produce a fully dynamic, non-linear simulation.

We combine FEM deformation with plastic deformations. Plastic deformations include tearing, fracturing, plastic stretching and cutting, but a full physically correct simulation of these phenomena is far beyond reach of the current hardware. We have taken a simplified model of cutting as a basic problem. The FEM operates on a mesh, so the basic form of cutting slices open the mesh at the position of a virtual scalpel, ignoring physical interactions between scalpel and material. In other words, the cutting problem boils down to cutting in meshes. The main problem with cutting in meshes is that tetrahedra are not closed with respect to cutting: the result of tetrahedron sliced with a blade is not another tetrahedron; the same holds for any other finite class of polygonal solids. Hence, the system has to contain routines that adapt the mesh to make cuts appear where a user performed them, while maintaining the tetrahedrality of the mesh.

Earlier work solved cutting operations in volumetric meshes by removing those parts of the mesh that came in contact with a surgical tool [9, 10], which is not realistic as it violates mass-preservation. In more recent work the basis is some deformable model defined on a tetrahedral mesh, with a subdivision scheme to accommodate cuts. In a subdivision scheme, tetrahedra that collide with a scalpel are subdivided to accommodate the scalpel. Bielser was the first to demonstrate fully volumetric cuts, albeit with an expensive scheme [14] that generated 17 tetrahedra for every split tetrahedron. It was later refined to be less expensive [15]. Other efforts include a dynamic level-of-detail model [16] and other subdivision schemes [17]. A common characteristic of subdivision schemes is that they increase the mesh size, which degrades the performance of the relaxation. Furthermore, injudicious use of subdivision may deteriorate the quality of the mesh, which can further hamper the performance of the relaxation.

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.

Cutting also plays a role in more distantly related work. For example, ChainMail [3] also supports cuts. Since ChainMail assumes a regularly structured grid, cutting operations are rather simple to implement, but like the surface of the object, the surface of the cut is always jagged to comply with the structure of the mesh. When the real-time restriction is relaxed, cutting is related to the field of fracture simulation, which has also found its way into computer graphics community [18].

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 subdivision can be computed, yielding an equation that relates deformation and elastic force. After the equations are specified, they can be solved with a variety of techniques. We support three-dimensional organs with arbitrary shapes. This can only be done with unstructured meshes, i.e., tetrahedral meshes.

2.1 Modeling elasticity

In this subsection, we briefly review linear elasticity. More thorough treatments can be found in texts on elasticity [19] and the FEM [20].

We have a tetrahedral mesh, and we assume that the displacement is piecewise linear on each tetrahedron, yielding the following equations for the gradient of the displacement $\mathcal{G}$

$$\begin{aligned} \mathcal{X} &= (\mathbf{X}_1 - \mathbf{X}_0 | \mathbf{X}_2 - \mathbf{X}_0 | \mathbf{X}_3 - \mathbf{X}_0)\,, \\ \mathcal{U} &= (\mathbf{u}_1 - \mathbf{u}_0 | \mathbf{u}_2 - \mathbf{u}_0 | \mathbf{u}_3 - \mathbf{u}_0)\,, \\ \mathcal{G} &= \mathcal{U}\mathcal{X}^{-1}. \end{aligned} \tag{1}$$

In these equations the displacements of node j for $j = 0, \ldots, 3$ are denoted by $\mathbf{u}_j \in \mathbb{R}^3$. Vectors $\mathbf{X}_j \in \mathbb{R}^3$ represent the undeformed location of node j for $j = 0, \ldots, 3$

The Lagrangian strain tensor $\mathcal{E}$ measures local deformation. It can be defined as

$$\begin{aligned} \mathcal{E} &= \frac{1}{2}((\mathcal{G} + \mathcal{I})^T(\mathcal{G} + \mathcal{I}) - \mathcal{I}) \\ &= \frac{1}{2}(\mathcal{G}^T + \mathcal{G} + \mathcal{G}^T\mathcal{G}). \end{aligned}$$

Here, $\mathcal{I}$ is the unit matrix. In the linear geometry approximation, we assume that deformations are small. Neglecting the quadratic term $\mathcal{G}^T\mathcal{G}$, we obtain the linearized relation for strain:

$$\mathcal{E} = \frac{1}{2}(\mathcal{G} + \mathcal{G}^T). \tag{2}$$

Both $\mathcal{E}$ and $\mathcal{G}$ are 2-tensors, i.e., matrices.

In the isotropic linear material model, we assume that the elastic energy density $W(\mathcal{E})$ is quadratic in the strain, and invariant under rotation. This assumption yields the following expression for W

$$W(\mathcal{E}) = \frac{1}{2}\lambda(\operatorname{trace}\mathcal{E})^2 + \mu \operatorname{trace}(\mathcal{E}^2).$$

The trace of a matrix is the sum of its diagonal elements. The numbers μ and λ are positive constants, called Lamé parameters. They represent elasticity properties of the material and are equivalent to the often used Young's modulus (denoted by E) and Poisson's ratio (denoted by ν).

By deriving the elastic energy of a tetrahedron with respect to $\mathbf{u}_j$ for $j = 1, 2, 3$, we obtain elastic forces on the nodes of a tetrahedron; these are denoted by $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$:

$$(\mathbf{f}_1|\mathbf{f}_2|\mathbf{f}_3) = \mathsf{volume} \cdot (2\mu\mathcal{E} + \lambda \operatorname{trace}(\mathcal{E})\mathcal{I})\mathcal{X}^{-T}. \tag{3}$$

Here, volume is the volume of the tetrahedron.

Finally, in a static deformation, every tetrahedron is in equilibrium, so the total force on a tetrahedron is the zero vector.

$$\sum_{j=0}^{3} \mathbf{f}_j = \mathbf{0}. \tag{4}$$

If we denote the force exerted by a tetrahedron τ on node v by $\mathbf{f}_{\tau,v}$, then the total elastic force on v is the contribution from all tetrahedra incident with that node:

$$\mathbf{f}_v = \sum_{\tau,\tau\ni v} \mathbf{f}_{v,\tau}. \tag{5}$$

As can be seen, this quantity $\mathbf{f}_v$ depends only on displacements of nodes that are connected to v by an edge, and the dependency is linear, since Equations (1)...(6) are linear. 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, or equivalently, when the energy of the system is minimal. In an equation, this is described as

$$f_{\text{external}} = -Ku. \tag{6}$$

In this equation f_{external} represents the total, global external force on the tissue. The vector u represents the global displacement of the tissue. Both are vectors of dimension $3n$, where n is the number of nodes in the mesh. Entries u_{3i+j} and f_{3i+j} contain the j-th component of displacement and force of node i respectively, where $i = 0, \ldots, n-1$ and $j = 0, \ldots, 2$. The matrix K combines all force-displacement relations in one big $3n \times 3n$-matrix called *global stiffness matrix*. In fact, every entry of the vector Ku is an elastic force that can be computed by Equation (5), which can be computed locally. In other words, Ku can be computed using only local information, i.e. without generating the matrix K itself. This locality is reflected by the sparsity of K: most entries of K are zero. Since K is the derivative of an energy function, it is positive definite.

2.2 Solving the equations

The equilibrium Equations (1)...(6) determine the solution up to a global translation and rotation. A unique solution is found by imposing additional boundary conditions. For example, the position of at least three vertices per connected component can be fixed. Combining these constraints, we get the following system of equations:

$$\begin{aligned} -Ku &= f_{\text{external}}, \\ u_{3i+j} &= \mathbf{p}_j^{(i)} \quad j = 0, \ldots, 2, \quad i \in \mathsf{fixnods} \end{aligned} \tag{7}$$

In this equation, fixnods is the set of fixed nodes, and $\mathbf{p}^{(i)}$ for $i \in$ fixnods are their prescribed positions.

The standard approach to solving this problem for instances with small n, is to substitute boundary conditions in K, thereby reducing its size, and attack the reduced K with standard numerical techniques such as Cholesky-decompositions. For larger instances, this is no longer feasible, and iterative methods are generally used. These methods do not rely on a decomposition of K, but compute a sequence $u^{(1)}, u^{(2)}, \ldots$ of approximations that converge to the solution of this problem. Memory requirements are kept low by exploiting the sparseness of K.

In our application, we continuously need a new solution: the simulation loops, continually responding to changing user input; for example, f_{external} and the boundary conditions may change continuously. For every change, we need an updated u. This suggests that we use an iterative algorithm, and always show the current iterand $u^{(k)}$ during operation. An even more compelling argument for iterative algorithms is that every time a surgical cut is effected, the size and value of K changes to reflect the updated mesh geometry and topology. In other words, the matrix K can change on-line, which precludes the use of any kind of matrix decomposition. We therefore chose to use an iterative method. To minimize the administrative hassle of updating K after cuts, we have chosen for a scheme where K is not stored explicitly.

As we have seen, every entry of Ku can be computed locally using Equation (5). This insight is the key to our solution method: we use an iterative method that only requires calculating Kd for some $3n$-vector d in every iteration. The iterative algorithms that have this property are called *Krylov*-subspace methods. We use the conjugate gradient algorithm [21]; it is the most popular algorithm of this class. It is especially suited for this problem since K is a positive definite matrix.

2.3 Convergence

We give a brief overview of the theoretical results of the convergence of conjugate gradients for FEM problems.

The convergence of a plain conjugate gradient iteration is linear, which means that the norm of residual, $\|Ku^{(k)} - f\|$, diminishes by a constant factor in every iteration. This factor depends of the condition number of K. The larger the condition number, the slower the convergence of the process.

- The larger the mesh, the slower the convergence. Typically, a conjugate gradient iteration for a FEM problem in three dimensions takes $\mathcal{O}(n^{1/3})$ iterations (i.e. the convergence rate is partially determined by the diameter of the mesh) [22].

- Material properties play a role. When $\lambda \to \infty$, then the compressibility of the material tends to zero. In an incompressible problem, K is singular, and an extra variable (pressure) must be introduced to solve the system. In practice this means that for $\lambda \to \infty$, the condition number goes up, and convergence deteriorates.

- The convergence is also influenced by the mesh characteristics. Poorly shaped elements—containing very small angles—increase the condition

number and hence deteriorate convergence rates [23].

2.4 Results

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 30 iterations per second. In our tests, the Lamé-material parameters are $\lambda = \mu = 1$ (which corresponds with Poisson's ratio $\nu = 0.25$). We declare convergence if $\|Ku - f\|/\|f\| < 10^{-3}$. For this material and this model, we observed that this model requires between 70 and 200 iterations to reach convergence.

On this machine, we found the conjugate gradient algorithm to be efficient enough for interactive use for models of around 1500 nodes. For models of this size, the solution does not appear instantaneous: 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 if this will detriment the perceived realism of a simulation. However, this effect does bound the maximal problem size: for significantly larger models, the cost of a single iteration is too high, and the solution process is visible as a jerky motion of the model.

We have observed the influence of compressibility in practice: the number of iterations required depends linearly on λ. The number of iterations did not become a problem in the instances we analysed.

We also experienced the influence of mesh quality. Cutting creates degenerate tetrahedra, as will be explained in Subsection 3.6. When these degeneracies are not removed, the convergence slows down dramatically. The number of iterations becomes an order of magnitude larger.

3 Cuts

A meaningful surgery simulation must support plastic operations on tissue. Unfortunately, most plastic deformations require much more analysis than the simple linear FE scheme we use. Therefore we concentrate on the operation with the simplest model available. We try to model cutting, where we ignore the physical interaction between scalpel and tissue, and only model the result of a cut: an incision.

Deformation makes cutting a little harder: modifications must be done to the original mesh, which represents the reference configuration, i.e. the undeformed configuration. However, the scalpel operates on the deformed model which is displayed. The shape of the mesh incision should match the scalpel surface in the deformed mesh. Then, after such a cut has been done, elastic relaxation might pull open the incision.

The cutting routine modifies the mesh, and the mesh also forms the basis for the relaxation scheme. A cutting method must therefore jibe with the deformation routines. In our case, we have a conjugate gradient relaxation for a FEM-induced system of equations. This puts forth the following requirements.

- The topology must remain consistent: the body must remain a manifold surface.

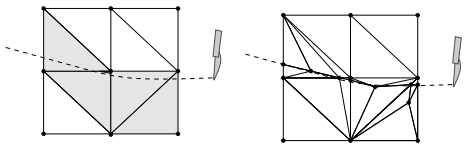

Figure 1: Cutting by subdivision. Active tetrahedra, those in contact with the scalpel, are subdivided. Degeneracies are inevitable if the scalpel passes close to mesh features.

- Elements must be connected by faces, if connected at all: the mesh should be a tesselation of a physically valid body. Such bodies do not allow connections without area.
- The mesh size should remain small, since the cost of a single relaxation step is proportional to the mesh size.
- The mesh should remain well shaped. This is desirable, since the convergence rate of an iterative scheme is highly dependent on the element shapes.

Previous solutions have had a preference for subdivision methods. In such methods, the chain of tetrahedra containing the scalpel are marked active. All active tetrahedra are subdivided to create an internal surface matching the scalpel sweep.

The advantage of this method is that the accuracy of the cut is high. Near the scalpel, the mesh has a high resolution. This means that the scalpel can be tracked accurately and promptly.

Subdivision also has disadvantages. A subdivision of a tetrahedron always requires at least four new tetrahedra. This means that enlargement of the mesh, and thus deterioration of interactive performance, is inevitable. Moreover, subdivision may create tetrahedra that are poorly conditioned. Experimental data on mesh quality in subdivision cutting schemes has not been presented yet, but we speculate that in situations such Figure 1, where the scalpel passes close to features of the mesh, badly proportioned tetrahedra are inevitable.

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 sub tasks, schematically shown in Figure 2.

1. Selecting faces to be cut from mesh. We call this process *surface selection*. It is discussed in Subsection 3.1.
2. Repositioning nodes from the mesh so that the selected faces are on the scalpel path. We call this process *node snapping*. It is discussed in Subsection 3.2.
3. Modifying the mesh to reflect the incision. We call this process *dissection*. It is discussed in Subsection 3.3.

4. Node snapping sometimes generates poorly shaped elements. Such elements hamper the relaxation so they must be removed. Dangling nodes caused by removing tetrahedra are also removed. We call this *degeneracy removal*. It is discussed in Subsection 3.6.

Finally, the static FEM analysis requires that every component has some boundary conditions, fixing the component. Cutting may create unconnected and unfixed components, so these have to be fixed.

3.1 Surface selection

The first step is to select which faces of the mesh are eligible for cutting. In other words, a set of triangles must be selected from the mesh. This set must be close to the path swept by the scalpel.

Let us define the scalpel more precisely. In reality, a scalpel is a small blade with one sharp edge. In a simulation, the scalpel will be a force-feedback device, which is sampled at some fixed frequency. By interpolating sampled scalpel positions we can obtain a sweep surface.

We use the following approach. Faces are selected on the basis of edge-scalpel intersections: for each tetrahedron, we intersect the sweep with all edges of the tetrahedron. From every intersected edge of the tetrahedron, we select the node closest to the intersection point. If the sweep intersects an edge twice, a consistent choice is made for either intersection point. The set of nodes that is acquired from one tetrahedron in this way generally has three or less points, and represents a feature of that tetrahedron. From every tetrahedron intersected by the sweep, a feature is selected. The union of all those features forms the cut surface. The procedure is demonstrated in 2D in Figure 3.

The surface that we select from the mesh must be close to the sweep, and its shape must resemble that of the sweep. Typically the scalpel sweep is a connected, non-branching surface, so the cut surface should also be connected and non-branching. We do not have proof that this approach satisfies all our criteria. However, the following observations suggest that the approach is satisfying in most cases.

- The cut surface is close the scalpel sweep, since features are only selected from tetrahedra where the sweep passes through.
- At most one triangle is selected from each tetrahedron; this limits the amount of triangles present in a cut surface, making extreme branching unlikely. Branching can and does happen, however. The dissect routine does handle this situation, but node snapping will typically flatten tetrahedra associated with the branch. This is one cause of degenerate elements.
- The cut surface is unlikely to be disconnected: if the sweep passes through two adjacent tetrahedra, these tetrahedra will also share the edge/sweep intersections of their shared face, so they will share parts of the cut-surface;

In the implementation, the sweep surface is assumed to be triangulated. This is convenient since calculating triangle/edge intersections is straightforward, and every sweep triangle intersects an edge exactly once. At most three nodes are

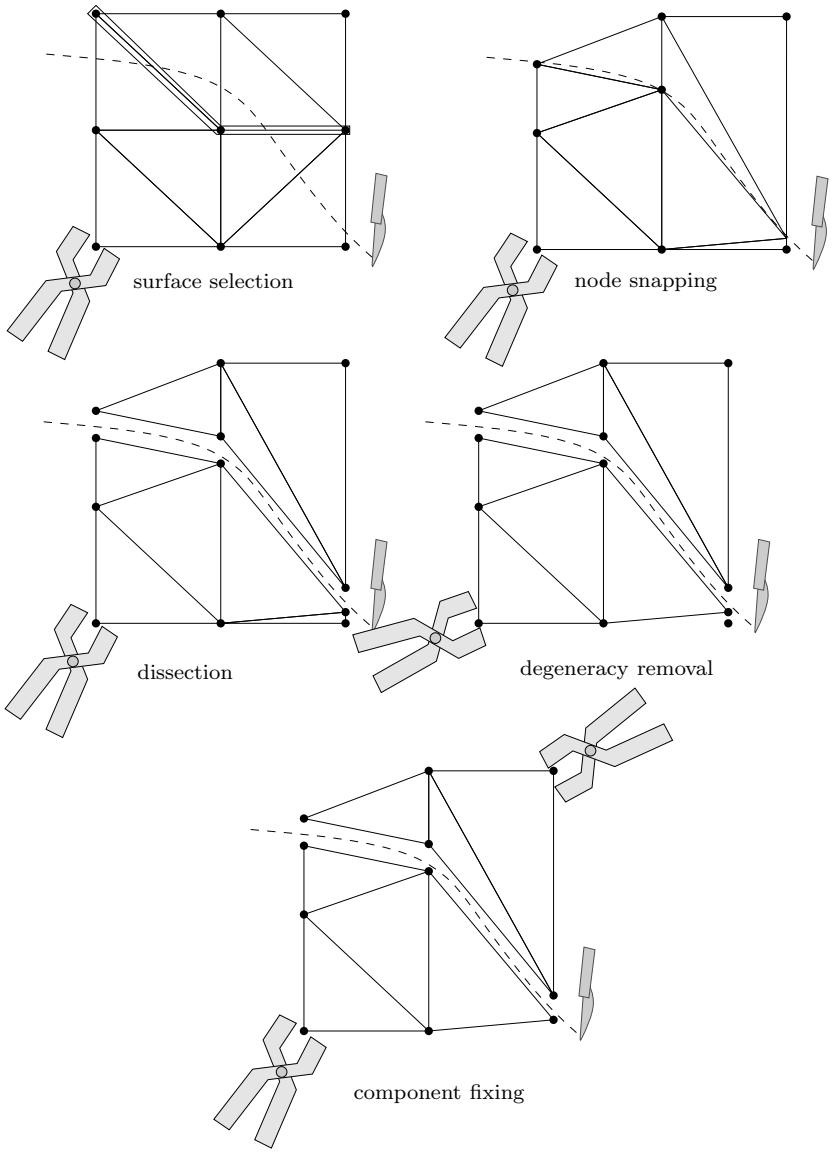

Figure 2: Steps in performing a cut, shown in 2D

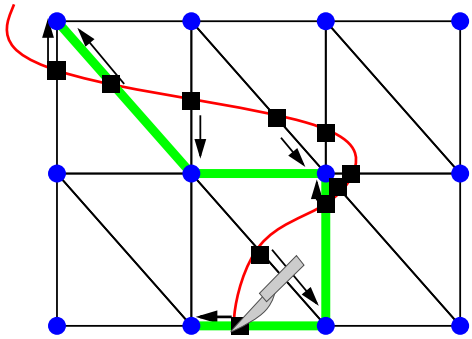

Figure 3: Surface selection using the closest node heuristic.

selected from every tetrahedron, except in the the degenerate case that the sweep triangle exactly halves four edges.

3.2 Node snapping

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.

To reposition a node v to some position $\mathbf{d}$, we must know τ, the tetrahedron incident with v that contains $\mathbf{d}$. The deformation of τ is used to transform $\mathbf{d}$ to the reference configuration. We proceed as follows.

- If v is an internal node, compute the new position $\mathbf{d}$ in the deformed mesh by projecting the deformed position of v orthogonally on the scalpel sweep.

  Find τ, the tetrahedron incident with v that contains $\mathbf{d}$. If no such τ exists, return failure.

- If v is a surface node, find all the surface faces incident with v. For each triangle Δ, project v orthogonally onto the intersection of the sweep and Δ. A projection is valid if it lies within Δ. Let d be the closest valid projection, and τ the tetrahedron whose boundary contains $\mathbf{d}$. If no such projection exists, return failure.

- The undeformed position $\mathbf{D}$ of $\mathbf{d}$ is computed using the deformation of tetrahedron τ.

- Move the node in the reference configuration to $\mathbf{D}$. Set the deformed position to $\mathbf{d}$. The subsequent relaxation step will move the node to its natural deformed position.

This method has been used for generating structured grids of arbitrary objects. In this application, a regular tesselation of an object is made by superimposing a regular grid on the object, and projecting nodes of the grid onto the boundary in the vicinity of the object [24].

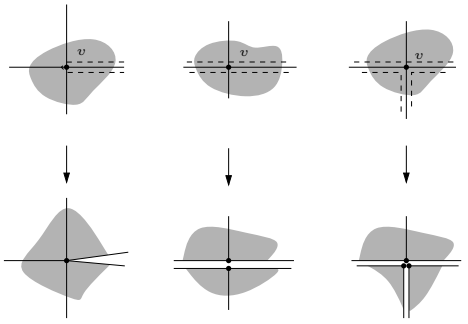

Figure 4: A cut surface can partition the set of triangles containing vertex v into one, two or more parts. The number of components (triangles connected by faces not in the cut surface) equals the number of nodes corresponding to v after the cut.

3.3 Dissection

When a set of triangles is selected from the mesh the mesh has to be modified to reflect the cut. This process is called dissection. Formally, this operation takes a tetrahedral mesh and a set of internal faces, called the *cut surface*.

If we single out a node v of the cut surface, we observe that dissection replaces v by a number of new nodes, as is shown in Figure 4. The cut surface divides the tetrahedra incident with v into a number of components, and dissection replaces v in every component with a new node. This suggests the following algorithm to perform the dissection. Given the cut surface C, it modifies the mesh to put triangles in cut surface C on the exterior boundary of the mesh.

Algorithm 1
1 for each node N within the mesh
2 $T :=$ the set tetrahedra incident to N
3 $K :=$ the set of components T is divided into by C
4 for each component c in K
5 $N_c :=$ a copy of N
6 substitute N with N_c in all tetrahedra of c.

This algorithm can only be conveniently implemented if the data structure for the mesh allows for easy substitutions. We have found a simple data structure that does this. We discuss it in the next section.

3.4 Mesh data structure

Data structures for meshes habitually represent meshes using object-oriented data structures: each mesh feature is an object, and if two features are incident, then this is represented by linking the two objects with pointers. Examples of such structures include radial-edge, facet-edge and triangle-edge [25–28]. Relying on objects and pointers makes it difficult to formally specify the data structure and provide provably correct modification operations.

For the FEM deformation we need topologically valid meshes. Moreover, both node snapping and surface selection do not give results with guaranteed

geometrical properties, so we can not rely on geometrical information when performing mesh operations. We have therefore chosen for a representation that explicitly represents only mesh topology, and represents that topology in terms of values, making it feasible to reason about them.

We describe the mesh using abstract simplicial complexes. This representation forms a basis when analyzing the nature of topological spaces [29]. We only use this representation as a basis for a data-structure.

When using simplicial complexes, each mesh feature (tetrahedron triangle, edge) is represented by an (abstract) simplex, i.e. the set of itself vertices. A simplex is a a signed sequence of nodes. The sign of a sequence represents the orientation of the simplex. Two sequences are considered equal if they are transformed into each other through an even permutation. Since any permutation is a composition of swaps, which are uneven permutations, we may as well define

$$\sigma[v_1, \ldots, v_k] = -[\sigma(v_1, \ldots, v_k)]$$

for any swap σ. Here, $[v_1, \ldots, v_k]$ denotes the oriented simplex associated with the sequence $v_1, \ldots, v_k$.

Through this definition we can obtain a unique representation. We can do this if the vertices are ordered: we can sort the sequence, while counting the number of swaps required. If l is this number, then a unique representation of $\pm[v_1, \ldots, v_k]$ is given by

$$\pm(-1)^l[\mathsf{sort}(v_1, \ldots, v_k)].$$

Simplexes in this standard representation may be ordered lexicographically. Thus, simplexes can be be used as a key in a search structures.

An inclusion relation is defined for simplexes. If $v_1, \ldots, v_k$ are vertices of the mesh, then

$$\mathsf{subsimplex}_j\ (\pm[v_1, \ldots, v_k]) = \pm(-1)^j[v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_k],$$

and a simplex p is contained in q if $p = \mathsf{subsimplex}_j q$ for some j. The *mate* of a simplex p is $-p$.

We say that tetrahedra τ_1, τ_2 are face connected if there are i, j such that $\mathsf{subsimplex}_i \tau_1 = -\mathsf{subsimplex}_j \tau_2$, i.e. if τ_1 and τ_2 have two orientations of the same face.

A tetrahedral mesh is cut regular 3-dimensional pseudo-manifold. Such a pseudo-manifold is a simplicial complex $\mathcal{K}$ formed by a set $\mathcal{T}$ of 3-simplexes (sequences with 4 nodes) and all their subsimplexes. We have three requirements for such a complex $\mathcal{K}$.

- The complex must be homogeneously 3-dimensional, which means that every $s \in \mathcal{K}$ is a subsimplex of some 3-simplex (tetrahedron) from $\mathcal{K}$.
- The complex must be a pseudo-manifold, which means that every oriented 2-simplex t from $\mathcal{K}$ (every triangle) is in exactly one 3-simplex of $\mathcal{T}$.
- The complex must be *cut-regular*. This means that for every pair of tetrahedra τ, τ' that both contain a vertex v, there has to be a chain of tetrahedra $\tau = \tau_1, \ldots, \tau_k = \tau'$ such that every τ_j contains v and every pair τ_j, τ_{j+1} is connected through some face.

An example of a very simple oriented 2-dimensional pseudo-manifold is given in Figure 5.

This mesh representation is convenient since it can be used to specify minimal, automatically verifiable requirements for a valid simplicial mesh. These requirements are minimal in the sense that they do not guarantee that a geometrically valid realization of the mesh exists. In fact, this problem seems generally intractable [30].

Notice that all the connectivity information of the mesh is present in $\mathcal{T}$ implicitly. However, the connectivity of the mesh is not directly available. This makes traversals relatively expensive. Moreover, the mesh features can not be identified across mesh modifications.

We therefore use a combined approach: mesh manipulation is done through operations on $\mathcal{T}$. For efficiency and identification purposes, tetrahedra and triangles are stored as object instances. Pointers between these objects are cached and can be used to speed up mesh traversals.

Both representations are linked: each mesh feature object (triangle, tetrahedron) has a field containing the simplex that it represents. The reverse association is maintained using a lookup structure (a hash-table or a balanced tree). As we have seen in the previous section, a dissection operation can be best described as an operation on simplexes. In this data structure, we can perform the operation on the simplexes. Using the updated simplexes, we can update the lookup structure. Using the updated lookup structure we can update the cached pointers. An example of the layout of the data structure is given in Figure 5.

3.5 Results

We have implemented this scheme, and succeeded in our goal: performing cuts without enlarging the mesh complexity. Figures 9 and 10 are screenshots from our prototype. They show the result of combining deformation and cutting.

The overall mesh quality is affected by cuts, as is shown in the graph in Figure 6. These graphs show how three tetrahedron quality measures are affected by cuts. Shown are histograms of the aspect ratio r, and the minimal dihedral angle d_{min}, and the complement of maximal dihedral angle d_{max}.

The aspect ratio r of a tetrahedron is defined as the smallest height divided by the maximum edge length:

$$r(\tau) = \frac{\min_{t \text{ triangle of } \tau} h_t}{\max_{vw \text{ edge of } \tau} |vw|}.$$

Here, h_t is the distance between a triangle t and the node opposite t.

The minimal dihedral angle is the smallest angle between two triangles of the tetrahedron

$$d_{min}(\tau) = \min_{s,t \text{ triangle of } \tau} \angle(s,t).$$

Likewise, the maximal dihedral angle measure is the largest angle between two triangles, subtracted from π.

$$d_{max}(\tau) = \pi - \max_{s,t \text{ triangle of } \tau} \angle(s,t).$$

These three measures are invariant under translation, rotation and scaling. Many more tetrahedron shape measures exist, and it is not clear which measure

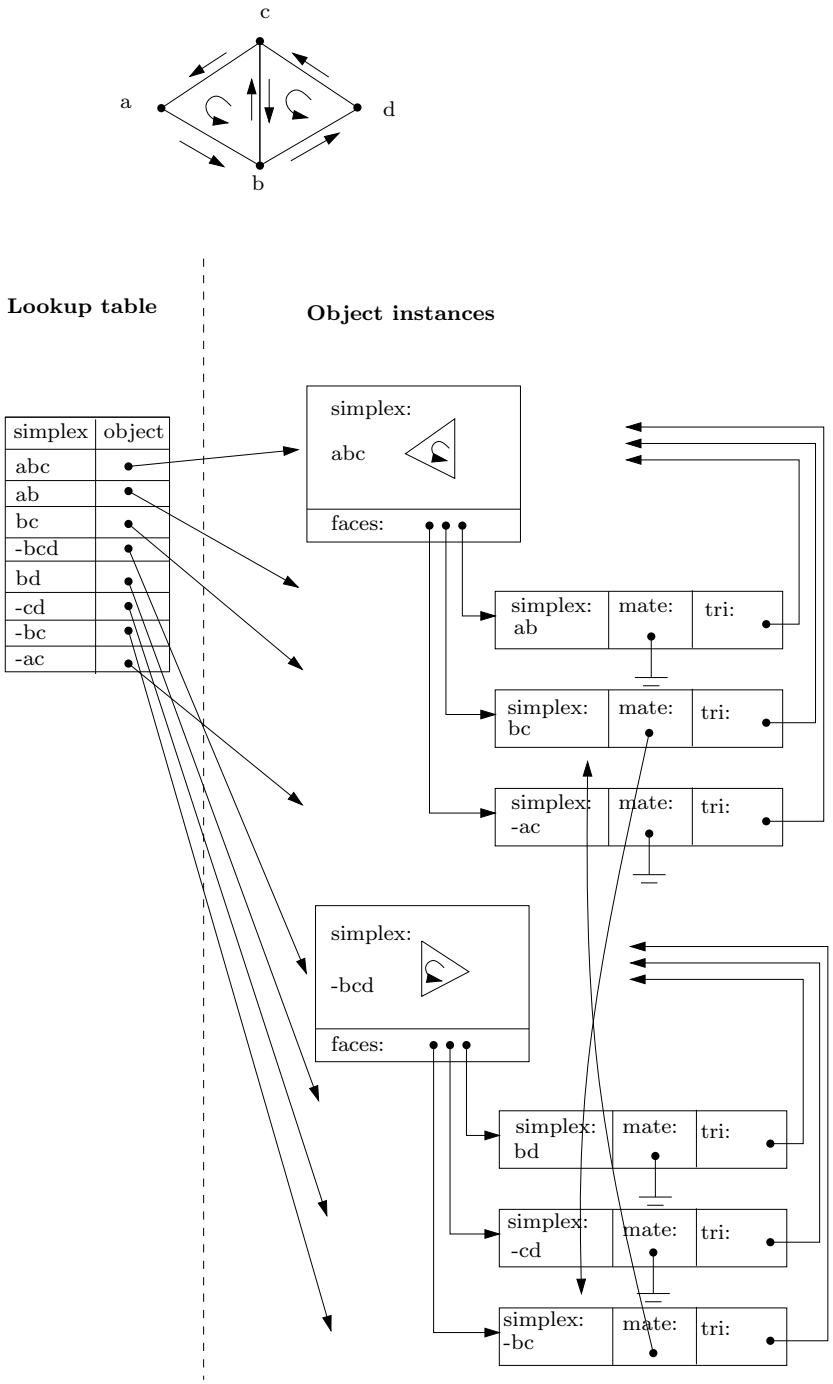

Figure 5: The 2D complex given by the complex $\{abc, -bcd, ab, bc, -ac, bc, cd, -bd, a, b, c, d\}$ with its representation in our data structure.

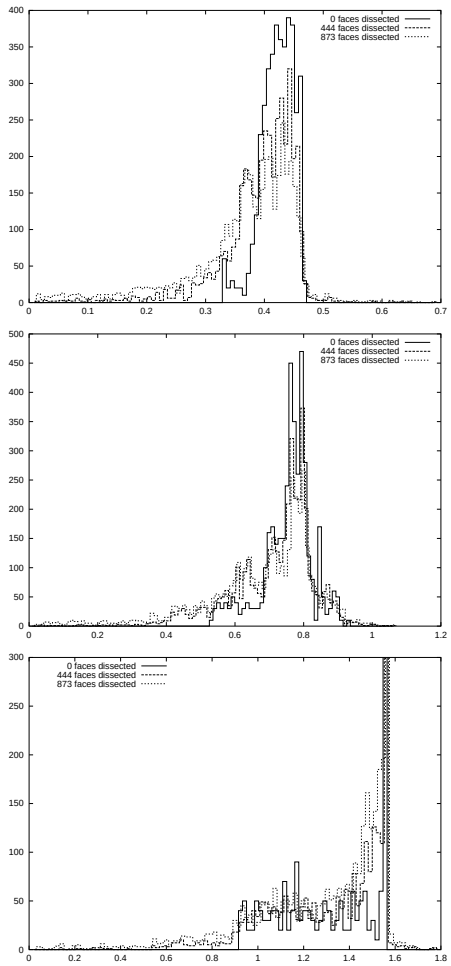

Figure 6: Histograms of tetrahedron quality measures after 0, 3 and 6 extremely large cuts. Shown from top to bottom are aspect ratio (minimum height divided by maximum edge length), minimum dihedral angle, π minus maximum dihedral angle. The cuts were done on a Delauney tetrahedralization of a cylinder (4020 elements, 891 vertices). 873 faces were dissected in total. 45 degeneracies were collapsed, where a degeneracy was defined to have an aspect ratio smaller than 0.01.

reflects the influence of mesh quality on FEM solutions best [31], however, all tend to 0 for badly shaped tetrahedra.

We can see that the mesh quality is affected by a cut. The diagram suggests that every cut spreads the distribution of tetrahedron shapes, mainly in adverse direction. However, we must note that the cuts were extremely large; more than half of the elements in the mesh were involved in a cut. The influence of smaller cuts is much smaller.

Although, we have succeeded in performing cuts without enlarging the mesh complexity, we have also encountered a number of other problems. Firstly, 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.

Secondly, 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 2, this happens with the upper left corner.

Finally, projecting nodes within the mesh can causes degeneracies. If four nodes of one tetrahedron are selected, node snapping will flatten that tetrahedron.[1] When all nodes of a tetrahedron are selected in the surface selection, then node snapping projects that tetrahedron onto a plane, giving a tetrahedron with zero volume and an aspect-ratio in the order of 10^{-17}. This situation is moderately rare. We found that the number of degeneracies were in the order of 5 % of the number of faces dissected, for our standard example, a Delauney tetrahedralization of a 891 node cylinder.

However, a degeneracy causes serious problems in the deformation routines, since a zero-volume tetrahedron responds to forces with infinite deformations. Therefore, degeneracies must be removed at all cost. This prompted us to implement a routine that removes these degeneracies. After each cut, all tetrahedra of the mesh are scrutinized, and tetrahedra with suspect aspect-ratios are removed.

3.6 Degeneracy removal

As we remarked, degenerate tetrahedra must be removed. Such tetrahedra can classified according to their shape: either a triangle of the tetrahedron is degenerate, or the tetrahedron itself is degenerate [32]. The degeneracies are listed in Figure 7, along with their removal strategies. Basically, in all situations, we try to identify an edge which is very short, and collapse that. If no such edge exists, we generate it by splitting the badly shaped tetrahedron appropriately. This mesh improvement is a heuristic approach, and solid proofs that these strategies work are absent. It shares this property with many other mesh improvement strategies [33].

We note that the edge-collapses are central in our removal procedure. An edge collapse of edge vw is in essence two operations applied: removing all tetrahedra incident with vw, and substituting a new node m for v and w in all tetrahedra incident with either v or w. The substitution can violate the manifold property, and removing a tetrahedron can violate the cut-regularity of any of its vertices. Hence the operation must be carefully checked before it is

[1] Snapping boundary nodes can also cause problems. An example is shown in Figure 2.

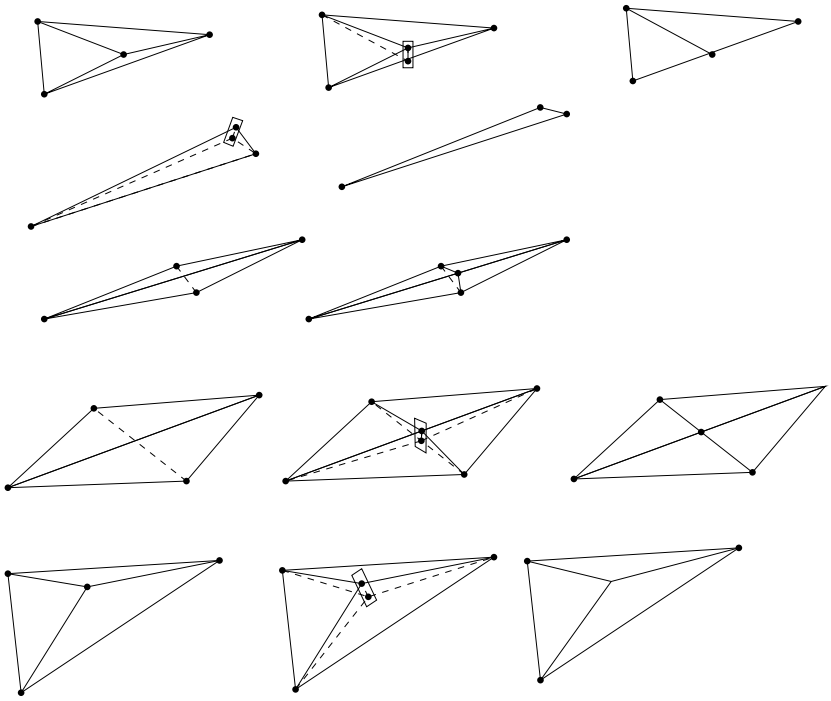

Figure 7: Strategies for collapsing triangles, needles, spindles, slivers and caps. Edges to collapse are boxed. A degenerate triangle is handled by splitting the longest edge, and collapsing the newly created diagonal edge. In a needle the shortest edge is collapsed. In a spindle, the edge opposite the flattest dihedral angle is split by a triangle. In a sliver, the two longest edges are split, and the newly created 'diagonal' edge is collapsed. In a cap, a node is added to the largest triangle, and the resulting diagonal edge is collapsed.

executed. In Figure 8, two examples of uncollapsible edges are given. The first one violates the manifoldness, the second one violates cut-regularity.

When an edge of a degeneracy can not be collapsed, the degeneracy can not be collapsed normally. If this is the case, we resort to removing the degeneracy by subjecting all its faces to a dissect operation. This cuts the degenerate element loose, causing a crack in the vicinity of the cut surface. When the element is loose, it is removed automatically. However, uncollapsible degeneracies are a rare circumstance, so we have no meaningful statistics.

In most cases collapsing degeneracies is sufficient, and we do not have to resort to removing degenerate tetrahedra. The degeneracy removal strategies also use subdivision, but fortunately, the number of tetrahedra only increases slightly due to this removal procedure. The experiment leading to the data in Figure 6, resulted in an increase of tetrahedra of 0.5 %.

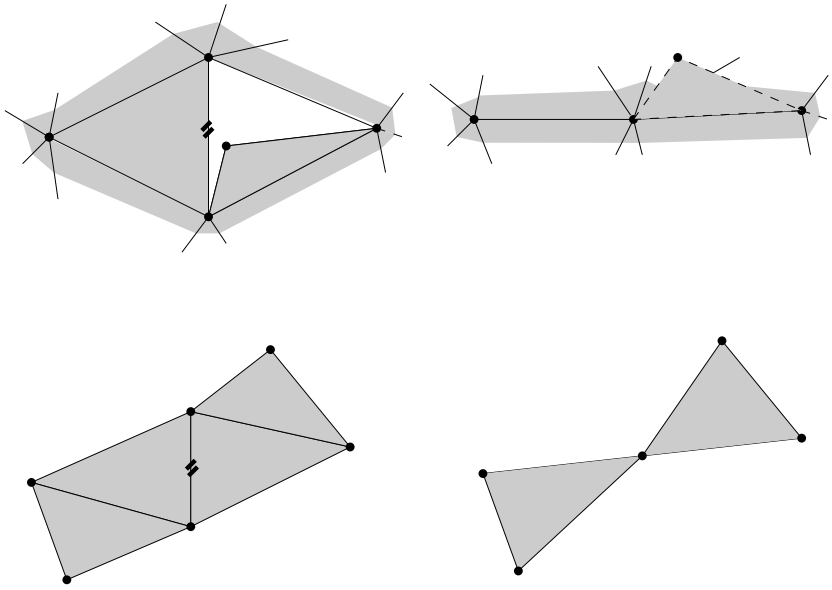

Figure 8: Two cases where an edge collapse fails. The to be collapsed edge is marked by ticks. In the top case, the manifold requirement is violated. In the bottom case, the collapse violates cut-regularity.

4 Discussion

We have made a twofold contribution: first, we have described how to implement a static linear FEM analysis without requiring precomputation, thus enabling interactive cutting. We have obtained satisfactory results, with deformations running at interactive rates for models with 1500 nodes. Second, we have demonstrated a cutting approach that does not increase the mesh size.

Like all previous contributions in interactive surgery simulation, we assume a number of idealizations in the deformation model. At present, our assumptions are 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 strain 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.

We plan to refine our approach to deformation to obtain a more realistic simulation. Our first subject of future research is the geometric non-linearity. Solving this requires more advanced numerical techniques. Solving one instance of a nonlinear problem involves solving several linear instances successively. The speed of a nonlinear solver thus crucially depends on the speed of a linear solver.

We intend to look into more advanced numerical methods such as multi-grid and preconditioning to speed up the convergence. Although combinations of such techniques have been used before in studies of parallel FEM computations [34], combining these techniques with interactive cutting is a new problem.

Our second contribution is 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, subdivision negates the original motivation 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] M. Bro-Nielsen, *Medical Image Registration and Surgery Simulation.* PhD thesis, Dept. Mathematical Modelling, Technical University of Denmark, 1997.

[2] H.-W. Nienhuys and A. F. van der Stappen, "Combining finite element deformation with cutting for surgery simulations," in *EuroGraphics Short Presentations* (A. de Sousa and J. Torres, eds.), pp. 43–52, 2000.

[3] S. F. Gibson, "3d chainmail: a fast algorithm for deforming volumetric objects," in *1997 Symposium on Interactive 3D Graphics*, pp. 149–154, ACM, 1997.

[4] M. A. Schill, S. 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)*, pp. 679–687, 1998.

[5] D. Terzopoulos and K. Waters, "Analysis and synthesis of facial image sequences using physical and anatomical models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 569–579, June 1993.

[6] M. Bro-Nielsen, D. Helfrick, B. Glass, X. Zeng, and H. Connacher, "Vr simulation of abdominal trauma surgery," in *Medicine Meets Virtual Reality 6*, (San Diego, California), pp. 117–123, IOS Press, 1998.

[7] P. F. Neumann, L. L. Sadler, and J. G. M.D., "Virtual reality vitrectomy simulator," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pp. 910–917, 1998.

[8] F. Boux de Casson and C. Laugier, "Modelling the dynamics of a human liver for a minimally invasive surgery simulator," in *Medical Image Com-*

puting and Computer Assisted Intervention (MICCAI), vol. 1679 of *Lecture Notes in Computer Science*, pp. 1156–1165, 1999.

[9] 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*, vol. 16, no. 8, pp. 437–452, 2000. to appear.

[10] 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.

[11] D. L. James and D. K. Pai, "Artdefo—accurate real time deformable objects," in *SIGGRAPH*, pp. 65–72, 1999.

[12] Y. Zhuang and J. Canny, "Real-time global deformations," in *Algorithmic and Computational Robotics*, (Natick MA), pp. 97–107, A. K. Peters, 2001.

[13] G. Székely, C. Brechbühler, R. Hutter, A. Rhomberg, N. Ironmonger, and P. Schmid, "Modelling of soft tissue deformation for laparoscopic surgery simulation," *Medical Image Analysis*, vol. 4, March 2000.

[14] D. Bielser, V. A. Maiwald, and M. H. Gross, "Interactive cuts through 3-dimensional soft tissue," in *Computer Graphics Forum (Eurographics '99)* (P. Brunet and R. Scopigno, eds.), vol. 18(3), pp. 31–38, The Eurographics Association and Blackwell Publishers, 1999.

[15] D. Bielser and M. H. Gross, "Interactive simulation of surgical cuts," in *Proc. Pacific Graphics 2000*, (Hong Kong, China), pp. 116–125, IEEE Computer Society Press, October 2000.

[16] F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno, "A multiresolution model for soft objects supporting interactive cuts and lacerations," *Computer Graphics Forum*, vol. 19, no. 3, pp. 271–282, 2000.

[17] A. B. Mor and T. Kanade, "Modifying soft tissue models: Progressive cutting with minimal new element creation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, no. 1935 in Lecture Notes in Computer Science, pp. 598–607, Springer-Verlag, 2000.

[18] J. F. O'Brien and J. K. Hodgins, "Graphical modeling and animation of brittle fracture," in *SIGGRAPH*, pp. 137–147, 1999.

[19] S. S. Antman, *Nonlinear Problem of Elasticity*, vol. 107 of *Applied mathematical sciences*. Springer Verlag, 1995.

[20] O. Zienkewicz and R. Taylor, *The Finite Element Method, The Basis*, vol. 1. Butterworth-Heinemann, 2000.

[21] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, Maryland: John Hopkins University Press, 1983.

[22] C. Johnson, *Numerical solution of partial differential equations by the finite element method*. cambridge university press, 1995.

[23] L. A. Freitag and C. Olliver-Gooch, "A cost/benefit analysis of simplicial mesh improvement techniques as measured by solution efficiency," Tech. Rep. ANL/MCS-P722-0598, Argonne National Laboratory, 9700 South Cass Avenue Argonne, Illinois 60439-4844, July 1998.

[24] J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., *Handbook of grid generation*, ch. 21, pp. 21–6–21–8. CRC Press, 1999.

[25] H. Lopes and G. Tavares, "Structural operators for modeling 3-manifolds," in *Solid Modeling '97*, 1997.

[26] M. Mäntylä, *An introduction to solid modeling*. College Park, MD: Computer Science Press, 1988.

[27] K. J. Weiler, *Topological Structures for Geometrical Modeling*. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1986.

[28] D. P. Dobkin and M. J. Laszlo, "Primitives for the manipulation of three-dimensional subdivisions," *Algorithmica*, vol. 4, pp. 3–32, 1989.

[29] F. H. Croom, *Basic Concepts of Algebraic Topology*. Undergraduate Texts in Mathematics, Springer-Verlag, 1978.

[30] I. Novik, "A note on geometric embeddings of simplicial complexes in a euclidean space," *Discrete & Computational Geometry*, vol. 23, pp. 293–302, 2000.

[31] A. Liu and B. Joe, "Relationship between tetrahedron shape measures," *BIT*, vol. 34, pp. 268–287, 1994.

[32] M. Bern and D. Eppstein, *Computing in Euclidian Geometry*, ch. Mesh generation and optimal triangulation. World Scientific, 1995.

[33] L. A. Freitag and C. Ollivier-Gooch, "A comparison of tetrahedral mesh improvement techniques," in *5th International Meshing Roundtable*, pp. 87–106, Sandia National Laboratories, October 1996.

[34] M. F. Adams, *Multigrid Equation Solvers for Large Scale Nonlinear Finite Element Simulations*. PhD thesis, University of California, Berkeley California 94720, January 1999.

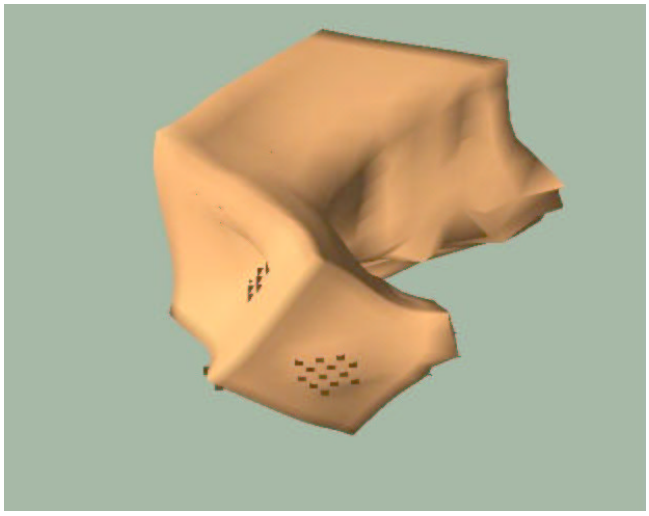

Figure 9: An incision in a $12 \times 12 \times 12$ cube (1728 nodes, 7986 tetrahedra). Nodes that have a fixed position are marked by ticks. Material parameters $\mu = \lambda = 1$. A dilating force is applied to the left and right. The last scalpel movement and the next scalpel movement are indicated by the two partially penetrating triangles. The deformation shown here is very large, and not accurate, due to the linear geometry assumption.

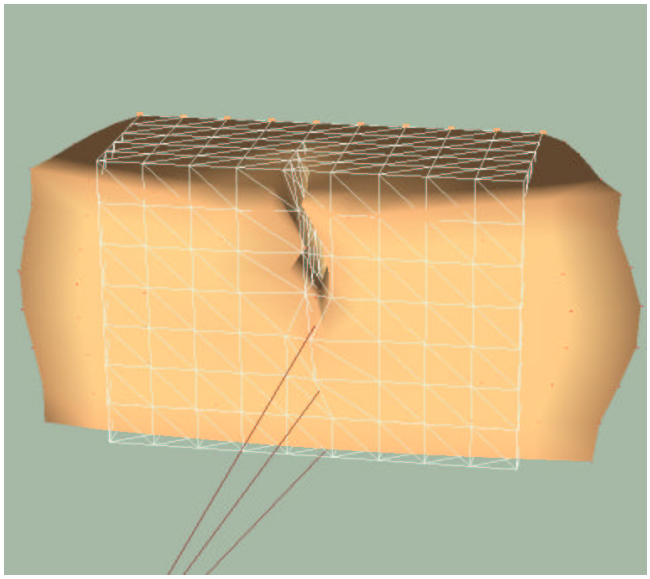

Figure 10: An incision in a mesh of 480 nodes. A dilating force is applied to the left and right side. The reference configuration of the mesh is also shown, and it clearly demonstrates the effect of node snapping. A little force has been applied to the incision to keep it from interfering with the original mesh, causing some dents in the incision surface-select.