Real-time Haptic Incision Simulation using FEM-based Discontinuous Free Form Deformation

Guy Sela[1,*] Jacob Subag[1,†] Alex Lindblad[2,3,‡] Dan Albocher[1,§] Sagi Schein[1,¶] Gershon Elber[1,‖]

Abstract

Computer-aided surgical simulation is a topic of increasingly extensive research. Computer graphics, geometric modeling and finite-element analysis all play major roles in these simulations. Furthermore, real-time response, interactivity and accuracy are crucial components in any such simulation system. A major effort has been invested in recent years to find ways to improve the performance, accuracy and realism of existing systems.

In this paper, we extend the work of [Sela et al. 2004], in which we used Discontinuous Free Form Deformations (DFFD) to artificially simulate real-time surgical operations. The presented scheme now uses accurate data from a Finite-Element Model (FEM), which simulates the motion response of the tissue around the scalpel, during incision. The data is then encoded once into the DFFD, representing the simulation over time. In real-time, The DFFD is applied to the vertices of the surface mesh at the actual incision location and time. The presented scheme encapsulates and takes advantage of both the speed of the DFFD application, and the accuracy of a FEM. In addition, the presented system uses a haptic force feedback device in order to improve realism and ease of use.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Visual

Keywords: Free-Form Deformation, Finite Element Model, Surgical Simulation

1 Introduction

Today, surgical simulators constitute an active research subject. Surgical simulators allow physicians to practice and hone their skills inside a virtual environment before entering the operating room. Such pre-operative training procedures have been shown to significantly improve the results of actual procedures [Seymour et al. 2002]. This is especially true with the recent increase in the use of endoscopic and laparoscopic procedures.

In order to maximize the potential gain in such virtual-reality training, a surgical simulation system should replicate the surgical environment as closely as possible in terms of look and feel. Conveying a realistic impression is difficult. Because of the complexity of such a task, it is best grasped when broken into smaller undertakings. One of the most important roles of any surgical simulator is to realistically animate - in real-time - the way tissue (skin and flesh or internal organs, etc.) behaves under cutting operations. A virtual cutting simulator should supply the following basic capabilities. First, it should have some mechanism for real time collision detection. Such a mechanism should control the location, direction and orientation of a virtual scalpel and constantly test for intersections with the model. Second, a cutting module should implement geometric operations that would progressively cut through the model, modifying its topology and constructing new geometry (the geometry of the cut) as needed and over time. Third, the cut model should reflect the physical behavior as accurately as possible, mainly presenting tissue behavior over time. Another important detail not to be overlooked is the user interface. A haptic force feedback device is invaluable in providing realistic interaction behavior, both from the visual and the palpable point of view.

When dealing with surface meshes, the actual task of cutting the tissue can be divided into two sub-tasks. First, there is the surface modeling task, in which the model surface should be split along the route of the scalpel as it advances. Second, the geometry around the cut should change, reflecting the shape and orientation of the cutting tool and the internal strain and stress properties of the tissue. In this work, we propose a framework that performs these two tasks. The framework is based upon an augmented variant of Free Form Deformation (FFD) [Sederberg and Parry 1986], which allows discontinuities and openings to be created in geometric models. The Discontinuous FFD (DFFD) [Schein and Elber 2005] is continuous everywhere except at the incision, and hence it has the ability to continuously deform the geometry around the cut. Moreover, we incorporate previously simulated results, using a Finite-Element Model, into the deformation function in order to make the behavior of the cut as realistic as possible.

Because FEM simulations are difficult to compute in real-time, an alternative approach could apply physical simulations to a low resolution representation of the model and encode it into the DFFD during the interaction, only to be immediately applied to the fine resolution representation of the geometry. This alternative approach would, of course, entail a much higher processing overhead, as the FEM simulation will need to be executed during run time, but on the other hand it will allow for more adaptable results than the first approach. In this work, we will concentrate on the first approach, in which the DFFDs are constructed off-line.

The proposed FEM-DFFD synergy is of low real-time computational complexity while retaining reasonable accuracy. Consequently, the algorithm is capable of handling complex geometric models at interactive rates. The FEM calculations are conducted

[*]guysela@cs.technion.ac.il
[†]jsubag@cs.technion.ac.il
[‡]alind@washington.edu
[§]sdannya@cs.technion.ac.il
[¶]sagi.schein@hp.com
[‖]gershon@cs.technion.ac.il
[1]Department of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel.
[2]Human Interface Technology Lab, University of Washington, PO Box 352142 Seattle, WA 98195-2142
[3]Department of Civil and Environmental Engineering, University of Washington, PO Box 352700, Seattle, WA 98195-2700.

once as a preprocessing stage using a straight-line scalpel path, whereas the deformation is applied to a limited local set of mesh vertices at every time step, mapping the straight path to a deformed path following the virtual scalpel.

The rest of this work is organized as follows. In Section 2, we give an overview of the previous work on the problems of cutting through and deforming geometric models. In Section 3, we describe the proposed cutting simulation approach; Section 4 presents a few examples and finally, we conclude in Section 5.

2 Related Work

Throughout the years, the problems of cutting through geometric models and deforming 3D models, for general as well as for medical purposes, have been tackled from many directions. In this section, and due to space constraints, we only consider a small subset of the relevant work. In Section 2.1, we look at work dealing with cutting through polygonal or tetrahedral meshes and in Section 2.2, we consider results related to the incorporation of FEM simulation results into medical simulations.

2.1 Mesh and Surface Cutting

Earlier work on mesh-cutting dealt mainly with surface-based meshes. Bruyns and Senger [Bruyns and Senger 2001] suggested a method of cutting polygonal meshes interactively without any post-processing, simply by splitting the affected polygons into several new polygons. A different approach proposed by Neinhuys and Van der Stappen [Nienhuys and van der Stappen 2004] suggested combining a local Delauny-based triangulation step as part of the mesh-cutting process. Edge-flip operations are used on the faces affected by the cutting operation in order to eliminate triangles with large circumferences. This method can be applied to both 2D and 3D surface meshes. A different approach, by Ellens and Cohen [Ellens and Cohen 1995], directly incorporated arbitrary-shaped cuts into tensor product B-spline surfaces. This approach has the advantage of operating over an inherently smooth surface, but requires modifications to the standard definition of trimmed B-spline surfaces.

Other works considered volumetric data models, mostly in the form of tetrahedral meshes. Using volumetric data models is beneficial as it can represent both the outer surface of the model as well as its inner parts. Ganovelli and O'Sullivan [Ganovelli and O'Sullivan 2001] proposed cutting tetrahedral meshes while re-meshing the tetrahedra around the cut to achieve the required level of smoothness. Since such splitting operations could degrade the quality of the mesh, they suggested using edge-collapse operations in order to remove low-quality tetrahedra from the mesh. Bielser et al. [Bielser et al. 1999] described cutting through tetrahedral meshes based on the observation that, topologically, there are only five distinct ways to cut a tetrahedron. Once the system detects a collision between a tetrahedron and the cutting scalpel, the case is mapped to one of the five available cutting configurations. The scheme uses a generic subdivision which replaces every original tetrahedron to be split with 17 new tetrahedra. This results in the introduction of many additional tetrahedra into the model and degrades the performance of the system over time. To circumvent this problem, Neinhuys and Van der Stappen [Nienhuys and van der Stappen 2001] proposed locally aligning the edges of the triangular faces around the cut to the route of the virtual scalpel. The movements of the scalpel inside a triangle are recorded and the vertices adjacent to the motion-curve are snapped onto it. Then, the triangles are separated along these aligned edges. Another approach, proposed by Forest et al. [Forest et al. 2002], treats cutting through tetrahedral meshes as a material removal problem. In [Forest et al. 2002], tetrahedra are removed from the mesh when hit by the pointing device. This trivially conserves the three-manifoldness of the tetrahedral mesh but results in a loss of mass of the volumetric model. Since the fineness of the cut is tightly coupled to the fineness of the model, this could result in sharp edges around the incision, something that is infrequently found when cutting human tissue.

2.2 Finite Element Deformation

Bro-Nielsen [Bro-Nielsen 1998] employed a linear elastic material model in order to gain speed at the expense of accuracy. The problem was that linear elastic models are only sufficient when dealing with small deformations. In another effort, by Mor and Kanade [Mor and Kanade 2000], model deformation was achieved by employing linear FEM over the cut model. Another problem that was tackled in [Mor and Kanade 2000] is the introduction of progressive cutting to prevent delays during the cutting procedure. Neinhuys and Van der Stappen [Nienhuys and van der Stappen 2000] tried to combine a FEM simulation by using an iterative solution to the set of equations with a conjugate gradient method. This method allows for alterations of the mesh topology at run time, as there is no preprocessing required. Vigneron et al. [Vigneron et al. 2004] take advantage of the XFEM method used in fracture mechanics to model cuts and resections in human tissue. XFEM does not require continuity within the mesh element and is thus useful for modeling cracks and cuts. Nonetheless, this can not be done in real time, as large systems of equations must still be solved during the actual simulation. Berkley et al. [Berkley et al. 2004] showed how constraints can be used to simulate suturing and support general real-time displacement-based interaction with finite element models. Wu and Heng [Wu and Pheng-Ann 2004] described a GPU-assisted system that uses coarse models to support limited real-time interaction.

Solving a large set of linear equations takes time. Even the use of iterative solvers is time consuming and rarely yields interactive frame rates. In our work, we incorporate the data from an off-line FEM simulation, trying to overcome the problems of solving these systems of equations at run time.

3 The Algorithm

The proposed algorithm operates in four phases:

- A first preprocessing stage. A FEM is created, simulating the cutting operation over time and in a canonical setup. Further, the locations of the individual elements are recorded at every time step. This stage is described in detail in Section 3.1.
- A second preprocessing stage, in which the data from the FEM simulation is encoded into a DFFD deformation model over time. This deformation is described in Section 3.2, and its encoding is discussed in Section 3.3.
- The real-time cutting stage. While the user is moving the scalpel along the skin, the skin polygons of the mesh are split in order to represent the cut. This step is presented in Section 3.4.
- The deformation stage. Following the cutting operation, the deformation is applied to the polygons around the cut, and

over time, splitting the cut open using the aforementioned deformation function. This stage is detailed in Section 3.5.

In addition, the specific support and integration of the haptic device into this simulation environment is described in Section 3.6.

3.1 Finite Element Model of Skin and Tissue

The finite element formulation used in this research has two separate element types: one for the skin, and one for the soft tissue below it. The skin element is a two-dimensional surface element that is bonded to the soft tissue. It has no stiffness in the transverse direction and is used to introduce tensile skin pre-stresses into the model. The soft tissue element is a volumetric element that smears the subdermal muscles, fat, tendons, etc. into one material representing the bulk behavior of the inhomogeneous continuum. This allows for a more accurate modeling of the skin and subcutaneous tissue, which inherently have very different material properties.

The finite element model is a three-dimensional volume that represents the patch of tissue being explicitly modelled. This volume is discretized with triangular constant strain elements and compatible prismatic wedge elements for the skin and soft tissue respectively. Along the sub-dermal boundaries that connect this volume to the rest of the body, distributed springs simulate the displacement-traction boundary conditions.

The cut path is defined before meshing occurs, and is used to define the shape of the mesh. Elements on the boundaries are aligned with the cut path and duplicate nodes are created along it. The duplicate nodes represent topologically distinct points on either side of the cut path. Without any additional constraints, the model represents the behavior of a tissue patch in which a cut has been introduced. Initial closure of the cut is imposed through the use of displacement constraints, as seen in Figures 1 and 2. These are algebraic constraints enforcing the condition that corresponding points on either side of the cut have similar displacements. As the virtual scalpel traverses the cut path, these constraints are relaxed, allowing the cut to open.

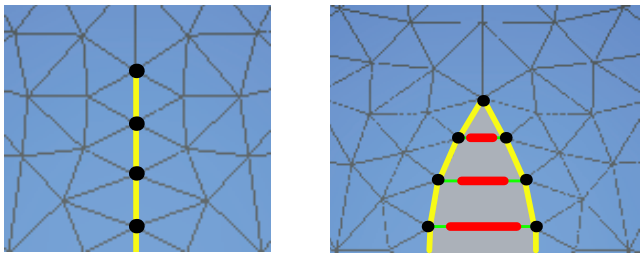

Figure 1: Physical and conceptual representations of the cut while the displacement constraints are active. Yellow (light) lines indicate locations of the cut whereas red (dark) bars indicate the actual displacement constraints.

3.1.1 Governing Equations

In general, the finite element equilibrium equations are written as $\mathbf{Ku} = \mathbf{f}$ where $\mathbf{K}$ is the stiffness matrix assembled from the stiffness matrices of the skin, soft tissue, and spring boundary elements, $\mathbf{u}$ is the vector of nodal displacements and $\mathbf{f}$ is the vector of externally applied loads. By adding to this system the constraints which model tissue separation at the cut path, the original system of linear equations is augmented to form the following 2×2 block system:

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f} \\ \mathbf{0} \end{Bmatrix}. \quad (1)$$

$\mathbf{C}\,\mathbf{u} = \mathbf{0}$ are constraints for enforcing closure of the cut. Each equation expresses the fact that two corresponding points on either side of the cut have the same displacements, in a given direction, when the cut is closed at that point. Algebraically, this may be written for two points A and B as:

$$\sum \phi_i(A) u_i - \sum \phi_i(B) u_i = 0, \quad (2)$$

where $\phi_i(X)$ is the value of the i-th finite element nodal shape function at location X, and the sum is over all nodal degrees of freedom in a given direction. The values $\phi_i(X)$ are the entries of the coefficient matrix $\mathbf{C}$. Given the compact support of finite element shape functions, $\mathbf{C}$ is very sparse. The vector $\mathbf{v}$ is a vector of Lagrange multipliers representing generalized forces necessary for enforcing the constraints (closing the cut).

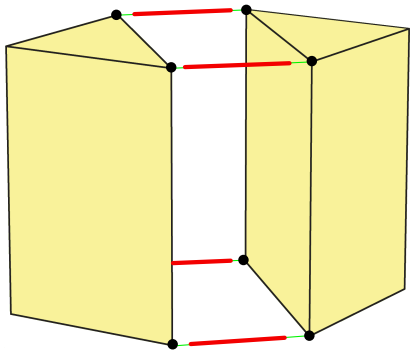

Figure 2: Conceptual image depicting the constraints in three dimensions. These are two elements that surround the cut, where the gap between them is part of the cut, but does not represent its full depth; which is typically modeled using multiple finite element layers. The linkages connecting the two elements represent the rigid displacement constraints that restrict the x,y and z displacements of the two nodes to be the same. Each set of linear prism elements are connected by four nodal constraints.

3.1.2 Solution Methods

Performing a block Gauss elimination on Equation (1) gives the following two equations,

$$\mathbf{C}\mathbf{K}^{-1}\mathbf{C}^T\mathbf{v} = \mathbf{C}\mathbf{K}^{-1}\mathbf{f} \quad (3)$$

$$\mathbf{K}\mathbf{u} = \mathbf{f} - \mathbf{C}^T\mathbf{v}, \quad (4)$$

where the solution of Equation (3) provides the values of the Lagrange multipliers and the solution of Equation (4) provides the displacement field. To advance the simulation as the tissue is cut, one set of displacement constraints, pertaining to the x, y, and z components of two attached nodes, is removed from $\mathbf{C}$ and Equations (3) and (4) are solved. This process is repeated until the cut is completed.

3.2 The Discontinuous Deformation Model

DFFD extends FFD to support the mapping of continuous domains into discontinuous ranges. Traditionally, FFD defines a mapping from $D \subset \mathbb{R}^3 \Rightarrow \mathbb{R}^3$, warping a region of $\mathbb{R}^3$ into another region of $\mathbb{R}^3$. When such mapping is applied to an embedded geometric object, $\mathcal{O}$, it also deforms its shape to follow the prescribed space warping operation.

FFDs may be defined using arbitrary functions $F : \mathbb{R}^3 \Rightarrow \mathbb{R}^3$. However, due to their robustness and controllability, trivariate tensor product B-spline functions are usually used. Such functions are defined as,

$$F(u,v,w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} P_{ijk} B_i^o(u) B_j^o(v) B_k^o(w), \quad (5)$$
$$(u,v,w) \in [U_{min}, U_{max}) \times [V_{min}, V_{max}) \times [W_{min}, W_{max}),$$

where P_{ijk} are the control points and $B_i^o(u)$ are the univariate basis functions of order o, in all three directions. As a result of using trivariate B-spline functions for deformation, a wealth of modeling tools, such as degree-raising and knot-insertion[Cohen et al. 2001], is made available.

As with any other modeling tool, FFD also possess several limitations. Specifically, FFD cannot support mapping into discontinuous ranges. Nevertheless, there are cases where the ability to model both the deformation and tearing of an object inside a unified framework, would be useful. To specify potential C^{-1} discontinuity into F, we use a standard knot-insertion procedure [Cohen et al. 1980]. By inserting order knots into $t = t_0 \in \{u,v,w\}$ parametric axis, a potential C^{-1} discontinuity is formed along this iso-surface. For brevity and without loss of generality, we will assume henceforth that knot insertion always occur along the v parametric axis at $v = v_0$ and that are no existing knots at $v = v_0$. As a result of inserting order knots into $v = v_0$, new control points are formed that interpolate the iso-surface $F(u, v = v_0, w)$. By manipulating the control points that are on or near iso-surface $F(u, v = v_0, w)$, a rich family of shapes can be modeled. In particular, for a virtual incision application the shapes of an arbitrary scalpel can now be realized.

FFD is commonly used to deform polygonal models. In such cases, the deformation is usually approximated by applying the deformation function, F, to the vertices of the model. However, in the case of DFFD, polygons that cross the discontinuity would not be mapped properly. To ameliorate this problem, the DFFD algorithm would split crossing polygons such that the edges of the polygon are clipped against the plane $v = v_0$. As a result of this clipping operation, new, non-crossing polygons replace the old crossing polygon. The splitting operation occurs in the parametric domain of F, hence the edges of a crossing polygon are always clipped against an axis-aligned plane, making the clipping operation much simpler.

One direct result of the above split operation is that closed models become open. Since for some applications this is an undesirable consequence, the DFFD algorithm should also supply means to seam the cut. For stability, vertices near the cut are translated in v by $\pm\varepsilon$ such that each is mapped to either side of the discontinuity. This operation only guarantees a C^0 continuity between the added geometry and the original one, at the incision location. In the next section, we will show how the results of an FEM simulation from Section 3.1 could be incorporated into the process of modeling the DFFD function.

3.3 Representing the DFFD Over Time

The result of the FEM simulation is a set of points $\{P_i^j\}$ where P_i^j is the location of point i, at frame j. Our interest lies in the volumetrically minimal subset that contains the points surrounding the cut from the first point at which the scalpel breached the skin, advancing along the cut, and until they reach a steady state, when the cut is fully open. We approximate the points' movement over time using a 4-variate (u,v,w,t) smooth B-spline function, which will represent the DFFD over time.

In our implementation, we used 4th order (cubic) B-splines and uniform open-end knot vectors, with the exception of the v-axis knot vector in which we added a discontinuity at the middle of the domain, simulating the cut (which is along the u-axis). This choice of orders, number of control points and knot vectors affects the level of accuracy. For example, increasing the number of control points (i.e., the degrees of freedom) would provide a DFFD that more accurately describes the results of the FEM simulation. A Least Squares (LS) fit problem could be defined which is linear in the 4-variate's control points' coordinates and corresponds to the following set of constraints:

$$\forall i,j \sum_{k,l,m,n} Q_{k,l,m,n} B_k(u_i^j) B_l(v_i^j) B_m(w_i^j) B_n(t_i^j) = P_i^j,$$

where $Q_{k,l,m,n}$ are the 4-variate's control points of indices k,l,m,n, $B_\alpha(\beta)$ are the α'th B-Spline basis functions of the selected orders and knot sequences evaluated at parameter value $\beta \in \{u_i^j, v_i^j, w_i^j, t_i^j\}$ of P_i^j.
The (u_i^j, v_i^j, w_i^j) parametric values of points $\{P_i^j\}$ were taken from the initial frame (i.e. the parametric values of point P_i^j are set to be the Euclidean coordinates of P_i^0), and the t parametric value is set to be j.

This technique's most significant drawback is evident in the case of regions in the deformed space that have too few sampled points and are thus underdetermined. For such regions, the LS algorithm results in control points being reduced to the zero point. In order to avoid such problems we modified the equation system's right-hand side to be $P_i^j - P_i^0$. Thus, the resulting control points' coordinates are actually the difference between the coordinates of the desired control points and the control points coordinates' values for an identity DFFD with identical orders, knot vectors and domain, meaning that the desired result is $I + Q$ where Q is the least squares result and I holds the control points coordinates values for an identity DFFD as specified above, i.e., $I(u,v,w,t) = (u,v,w,t)$. The resulting DFFD, and following the off-line FEM simulation, continuously describes the way the volume of the canonical tissue surrounding the cut deforms over time. An example is depicted in Figure 3.

3.4 Splitting the Geometry

DFFDs can be applied to surface meshes, volumetric data sets and parametric models. For the sake of efficiency, we focus our work on triangular surface meshes. When cutting through a triangular mesh, the most basic operation is triangle splitting. Any triangle that has vertices on both sides of the cut must be split before the DFFD mapping is applied to it. Moreover, vertices on the cut line must be treated with care, as will be discussed shortly. This splitting operation is conducted incrementally as the virtual scalpel advances, one triangle at a time, following the path of the cut, along the skin surface.

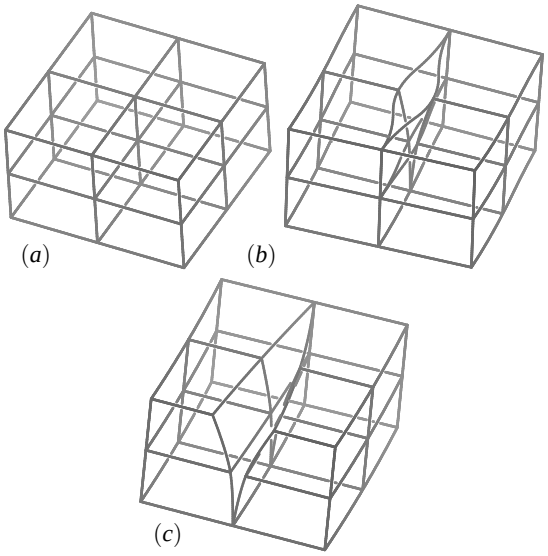

Figure 3: Sampling of the 4-variate DFFD at $t = 0$ (a), $t = 0.2$ (b) and at $t = 0.8$ (c).

The following notations will be used: Let $C(s)$ be an arc-length parametric representation of the cut line following the path of the virtual scalpel. Similarly, $N(s)$ defines the orientation of the virtual scalpel at every time t.

Usually, splitting a triangular face creates three new triangles; one triangle on one side of the cut, and two on the other side (see Figure 4). These three new triangles replace the original face, which is then purged. These original triangles have both entry and exit cut locations, found by calculating the closest point on a segment (the entry or exit edge) to $C(s)$. In addition, we examine how close this linear approximation of the cut line is. If the straight line between the entry and exit locations is sufficiently close to $C(s)$, that triangle is split as just described. Otherwise, the triangle must be subdivided recursively into smaller triangles before the split can take place. Figures 5 shows an example of this more complex case.

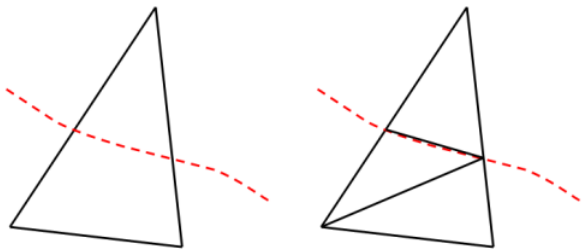

Figure 4: Triangle split, normal operation. The dashed line indicates the cut path.

If the cut passes very close to a vertex, we handle things a bit differently, as the splitting method that was proposed above will create one very small triangle and one very long and thin triangle. In this case, we move the vertex at hand onto the cut, duplicate it, and use one copy on each side of the cut, see Figure 6.

While we only process the skin surface, we also seek to model the deepness of the cut, and model this new geometry on the fly. In order to do this, the entry and exit locations are duplicated a number of times and moved in the direction of $-N(s)$ into the body. Triangles are then used to tessellate these interior vertices, all the way to the bottom of the cut. The resulting geometry can be seen in Figure 7.

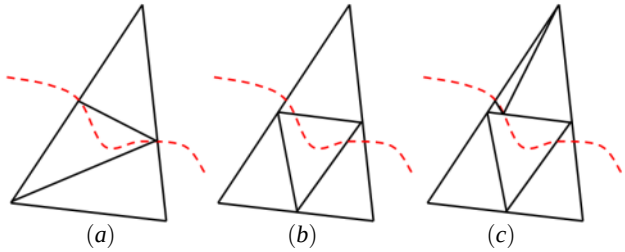

Figure 5: When the straight line between the entry and exit points is not close enough to the cut path, the triangle is subdivided and the algorithm continues. (a) shows the naive split, (b) shows the subdivision, and (c) shows the first child triangle after the split.

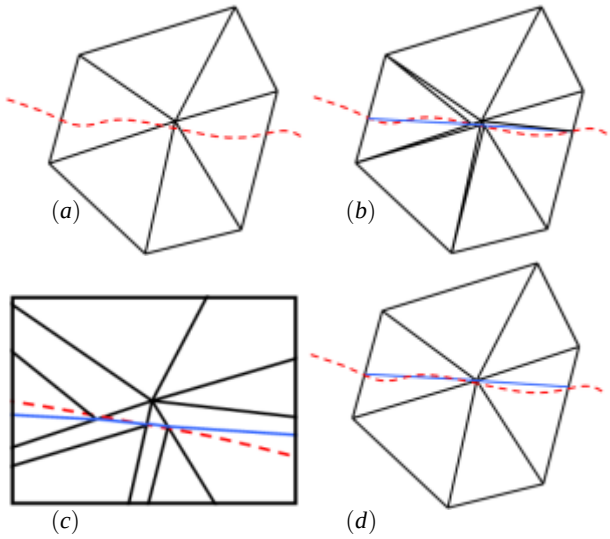

Figure 6: When the scalpel path lies very close to a vertex, the geometry is split in a different manner. The dashed line indicates the scalpel path. (a) shows the initial configuration. (b) shows the splitting of the neighborhood triangles according to the basic algorithm, (c) is an enlargement of the center vertex area. (d) shows our modified solution for the problem, which moves the original vertex to the path and splits only two triangles.

3.5 Time Dependant Deformation of the Geometry

The DFFD is applied at regular time intervals to a list, $\mathscr{L}$, of active vertices around the cut. The active vertices in $\mathscr{L}$ are vertices of the original mesh near the cut, in addition to the vertices created when the triangles around the cut were split. A vertex is entered into $\mathscr{L}$ when the scalpel passes near it (in the case of original vertices) or when it is created (for the new ones), and removed from $\mathscr{L}$ when it is no longer affected by the DFFD application (as is detailed later on). At every time step, $\mathscr{L}$ is updated, and the scalpel path ($C(s)$) and scalpel orientation curve ($N(s)$), are recreated. From these curves and the width of the cut, we reconstruct the incision parametrization volume (IPV), see Figure 8. The IPV is actually the

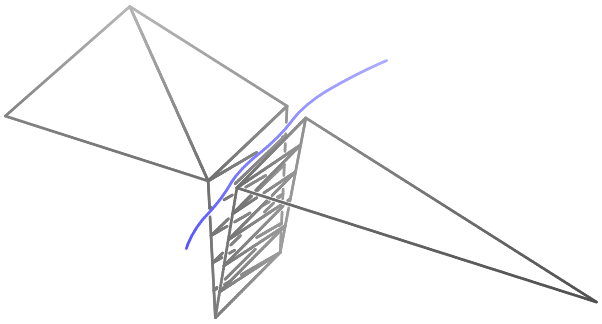

Figure 7: The deepness of the cut polygons modelled and added to the mesh.

volume inside of which reside all the vertices of $\mathscr{L}$, or vertices that will be moved as a result of the next DFFD application. The IPV's role is to correlate between a vertex's Euclidean coordinates and its coordinates in the (canonical) parametric domain of the DFFD.

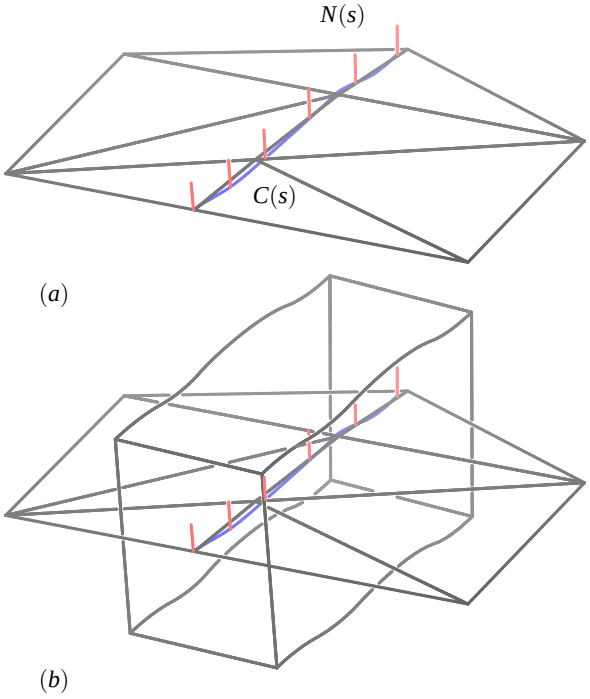

Figure 8: IPV construction. The volume is constructed along the curve, appearing with respective scalpel orientations at regular intervals, as seen in (a). The volume is seen in (b).

The vertices in $\mathscr{L}$ are mapped through the IPV, creating the u,v,w and t coordinates for the DFFD. For every vertex V_i in $\mathscr{L}$, let $v_s = \arg\min_s \|C(s) - V_i\|$. Then, u and w are set by their distance to $C(v_s)$ along $N(v_s)$ and $T(v_s) \times N(s)$, respectively, where $T(s) = C'(s)$. The t parameter is, again, the time in which the vertex was entered into $\mathscr{L}$. Since only parameter values of new vertices, that enter $\mathscr{L}$, need to be recomputed for every DFFD iteration, the procedure is quite efficient.

Hence, we have $(u',v',w') = DFFD(u,v,w,t)$ $= DFFD(IPV^{-1}(x,y,z),t)$. We then apply the IPV to these (u',v',w') parametric coordinates in order to find the new Euclidean location of the vertex.

A vertex $V_i \in \mathscr{L}$, inside the volume surrounding the cut, will move an amount that is the sum of all the small deformations assigned to it in all the iterations of all DFFDs, while V_i is in the incision volume.

3.6 Force Feedback Support

In order to provide a more realistic use for the proposed approach, we enabled a SensAble(tm) PHANToM Desktop(tm) haptic device [SensAble Technologies, Inc.] to work with the system. see Figure 9. The device consists of a pen-sized handle connected to a robotic arm with flexible engine-enforced joints. These allow the arm to move in 6 degrees of freedom (DOF): three spatial coordinates and pitch, roll and yaw. The engines at the arm joints allow the device to apply force to the holder of the handle, providing it with force feedback ability. This pen-sized handle was used as scalpel in our simulation, so when adjusting the haptic resistance correctly, the combination of the visual and haptic user interface provided us with a convincing look and feel of an actual incision process.

Figure 9: PHANToM Haptic Device in operation.

Due to the sensitivity of the human tactile system, a force feedback device requires update rates of 1kHz in order to provide high definition simulations (see [Tan et al. 1994]). Therefore, our algorithm must handle collision queries between the virtual scalpel and model at these rates. A brute force algorithm, which iterated through all polygons of the model and performed collision tests with the scalpel, resulted in poor frame rates with models of the order of tens of thousands of polygons. Therefore, to speed up the collision detection process, we preprocessed the model data, and created a uniform voxel grid around the model, where each voxel holds a list of polygons that intersect the voxel. This limited our collision queries to only the polygons in the voxels intersecting the scalpel, and effectively reduced our calculations to a few hundred line-polygon intersection tests at most, for every collision query between the virtual scalpel and the model.

Our force feedback model has two stages: a pre-puncture stage and a post-puncture stage. When light forces are applied to the skin, a force model is used to simulate the scalpel touching the skin, without penetrating it [Terzopoulos et al. 1987]. In this model, the force is a function of the depth to which the scalpel is pushed in the direction normal to the skin. When the scalpel pierces the skin, this force is replaced by a viscous drag force (according to Stokes'

model) $-bT(t)$, [Terzopoulos et al. 1987] (where b is an approximated constant of viscosity extracted from experiments and $T(t)$ is the the speed vector of the scalpel's cutting path). This simulates the movement of the scalpel while cutting through skin or flesh. Constants were chosen empirically, to provide realistic touch and cut force feedback. Eventually, the 6DOF control over the scalpel, along with the force feedback feature, provided us with intuitive control over the scalpel, and raised the simulation to a higher level of realism.

4 Results

Figures 10, 11 and 12 show snapshots from a few incision simulations generated with our implementation. In Figure 10, an incision simulating a brow-lift is shown, Figure 11 shows a typical incision made in a face-lift operation and Figure 12 shows a side view of an incision being made across the bridge of the nose, displaying a side view of the cut.
The simulations were performed on a P4-2.8GHz desktop computer with 1GB of RAM. The original head model consists of 12108 polygons. The cuts shown in Figures 10, 11 and 12 added 453, 1443 and 1082 polygons, respectively, including the polygons representing the deepness of the cut.

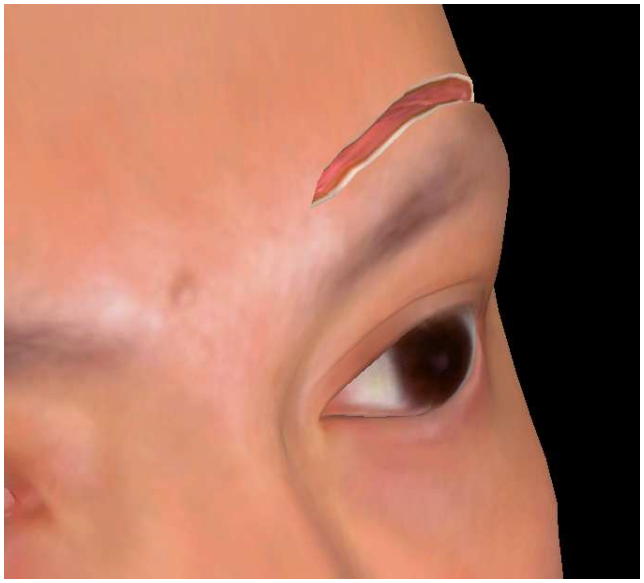

Figure 10: A brow-lift cut simulation.

Figures 10,11,12 and 13 are all from a real-time film recording of the actual use of the force feed-back scalpel. The full incision video and other results are available at:
http://www.cs.technion.ac.il/∼guysela/incision.htm

5 Conclusions and Future Work

We have presented an enhancement to our previous work [Sela et al. 2004] in which we proposed a method to perform real-time incision simulation using 3D DFFDs. In this enhancement, we detail how to compute an incision simulation using an off-line FEM simulation and incorporate this simulation's results into a 4D DFFD representation over time. Finally, we demonstrated how this 4D DFFD can

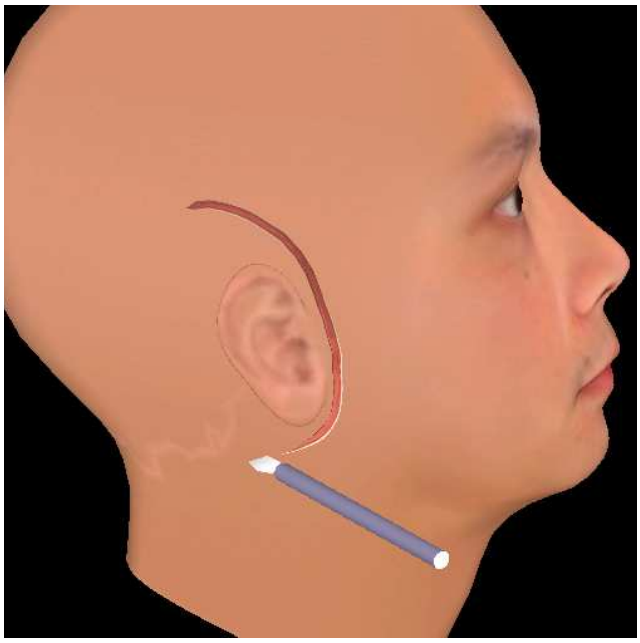

Figure 11: A face-lift cut simulation.

be used to simulate real-time incisions on a 3D model using a haptic device. This method is modular and, therefore, flexible enough to accommodate different methods of incision simulation as well as different model representations. It is also computationally simple enough to yield real-time frame rates on desktop computers.

Future extensions may include the usage of more than one simulation results, i.e., using different FEM simulations and corresponding DFFDs for different types of incisions, handling incisions with variable width and depth, depending on material properties, and the application of the method to volumetric data. Another possible extension could be the exploration of different surface mesh cutting techniques. Our system uses a recursive, non-adaptive scheme, and an adaptive one could reduce the number of polygons added to the model at the expense of additional computation.

Furthermore, the presented framework can be extended to model different cut shapes, representing bent scalpels, and altogether different deformations of the model such as bend, protrude, twist, etc., functioning as a cutting modeling tool.

Another potential use of the technology presented in this paper is to convey the results of large scale finite element models to scientists and engineers. Frequently numerical models in many important application domains such as structural mechanics simulations, wave propagation in heterogeneous media, weather prediction, etc. generate multiple, large-scale 3D data sets. The multiple data sets arise from simulations that are run with different parameters, discontinuities, boundary conditions, interfaces, etc, in order to fully quantify a phenomenon or evaluate a design. An important task for the scientist/engineer user is to browse the massive data generated from these simulations with the goal of finding critical patterns and features and assessing changes in the model response due to changes in loading, size of cracks, contact regions and a myriad of application-specific model parameters. The task is tedious and time consuming and in the current state of the art often involves either a round-trip call to the underlying finite element simulation, or manual access and parsing of data files from a separate storage subsystem. This prevents real-time, free style, exploration of the model response. Further, and perhaps more importantly, this *staccato* mode of ex-

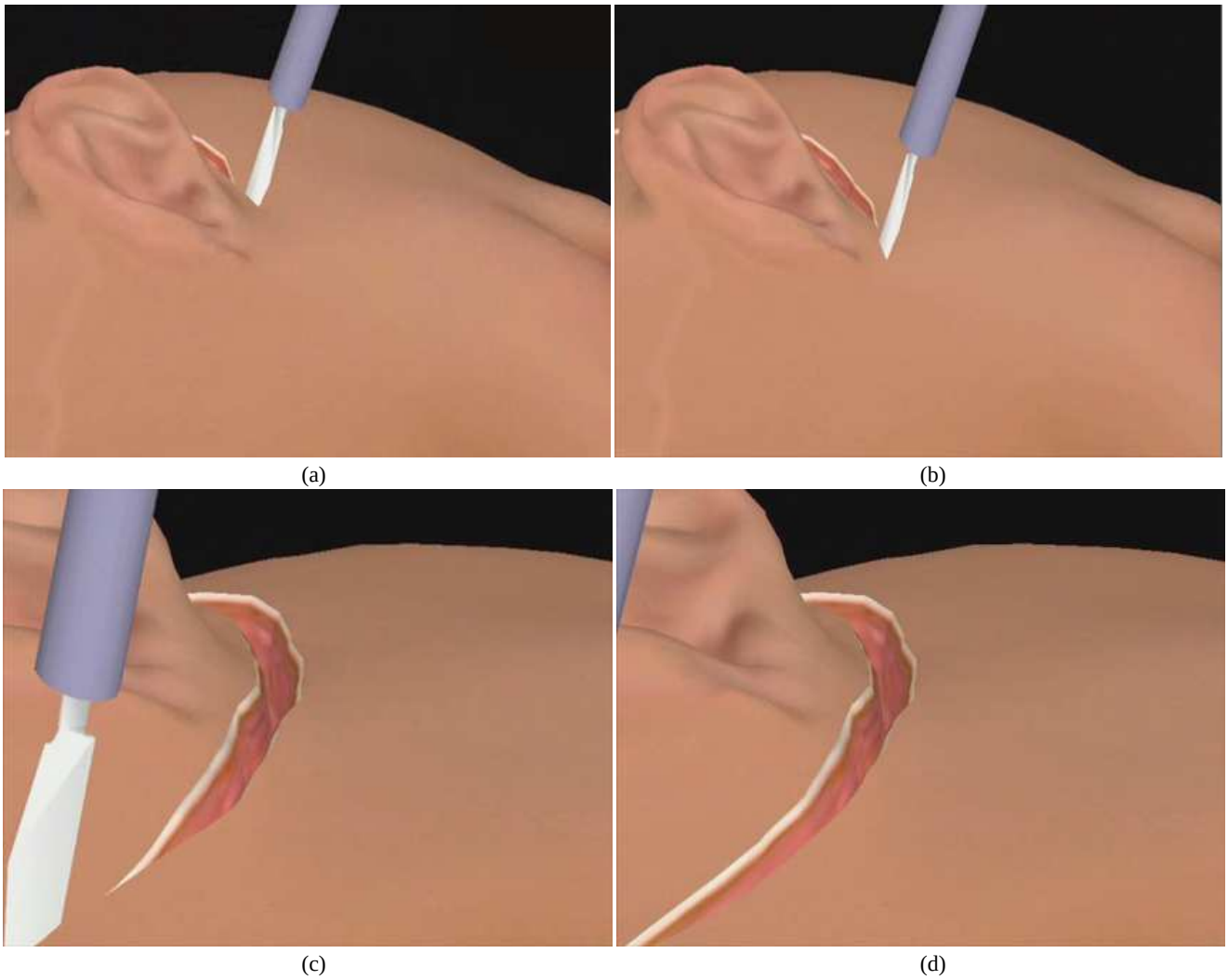

Figure 13: Four snapshots of the interactive incision process of the face-lift simulation.

ploring response data is, by its nature, a hinderance for understanding how the response is affected by changes in model structure, topology, boundary conditions and other parameters. Discontinuous FFD presents a practical mechanism for storing finite element data sets with discontinuities and conveying them effectively to end users. We have shown that by using DFFD, even data sets with spatial discontinuities can be encoded and then replayed, rendered and manipulated at interactive speeds. This allows users to explore the data produced by the numerical simulations and quickly find the specific spatial regions, combination of parameters, or critical lengths of contacts/discontuities, etc. that are of most interest. This information can then be used to access portions of the underlying raw data for additional, more detailed analysis. DFFD provides the ability to essentially summarize and capture large amounts of finite element data sets, while allowing users to visualize and browse warped data, possibly with discontinuities and then ”zoom in” to access specific data items at interactive rates. This is a key supporting technology for the development of scientific computing applications.

One drawback of the system is the inability to deal with more complex situations, such as large lacerations, tissue removal or the creation of a skin flap. Another limitation is due to the fact that our FE model assumes that the entire simulated volume is comprised of tissue without the presence of, say, bones. As the framework performs the FEM calculation only once, it is not possible to encode such information into the FFD since the location and orientation of a bone is unknown during preprocessing. One more noteworthy point is the fact that although we used a multivariate B-Spline function for encoding the FEM data, it is by no means the only method. Any other representation that captures the simulation data over time would suffice as long as it can be retrieved quickly enough when required in real-time for fast calculation of the vertices’ new locations.

6 Acknowledgments

This research was support in part by the Israeli Ministry of Science Grant No. 01-01-01509, in part by European FP6 NoE grant 506766 (AIM@SHAPE), and partially by the fund for promotion of research at the Technion, Haifa, Israel.

The third author would like to thank the Department of Energy’s Computational Science Graduate Fellowship, and the Krell Institute, for funding a portion of this research.

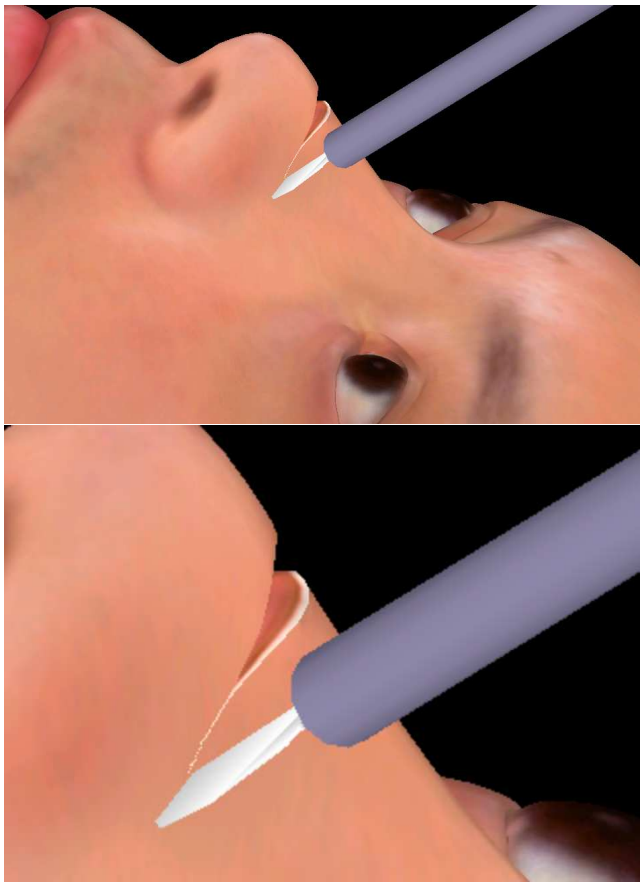

Figure 12: Rhinoplasty cut simulation (side view). Bottom image presents a zoom of the relevant area of the top image.

The face model was retrieved from the 3D-cafe web site, www.3dcafe.com. All the illustrations were prepared with the aid of the Irit solid modeling system, www.cs.technion.ac.il/∼irit.

We would like to thank Jihad El-Sana who allowed us to use his PHANToM(tm) device for the duration of this project.

References

BERKLEY, J., TURKIYYAH, G., BERG, D., GANTER, M. A., AND WEGHORST, S. 2004. Real-time finite element modeling for surgery simulation: An application to virtual suturing. *IEEE Trans. Vis. Comput. Graph. 10*, 3, 314–325.

BIELSER, D., AND GROSS, M. H. 2000. Interactive simulation of surgical cuts. In *Proceedings of Pacific Graphics 2000*, I. C. S. Press, Ed., 116 –125.

BIELSER, D., MAIWALD, V. A., AND GROSS, M. H. 1999. Interactive cuts through 3-dimensional soft tissue. In *Computer Graphics Forum (Eurographics '99)*, P. Brunet and R. Scopigno, Eds., vol. 18(3), 31–38.

BRO-NIELSEN, M., AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum 15*, 3, 57–66.

BRO-NIELSEN, M. 1998. Finite element modeling in surgery simulation. *Proceedings of the IEEE 86*, 3, 490–503.

BRUYNS, C., AND SENGER, S. 2001. Interactive cutting of 3d surface meshes. *Computers & Graphics 25*, 4, 635–642.

CAMARA, O., DELSO, G., BLOCH, I., AND FOEHRENBACH, H. 2002. Elastic thoracic registration with anatomical multiresolution. In *International Journal of Pattern Recognition and Artificial Intelligence*, World Scientific. Special Issue on Correspondence and Registration Techniques.

COHEN, E., LYCHE, T., AND RIESENFELD, R. 1980. Discrete bsplines and subdivision techniques in computer aided geometric design and computer graphics. *Computer Graphics and Image Processing 14*, 87–111.

COHEN, E., RIESENFELD, R. F., AND ELBER, G. 2001. *Geometric modeling with splines: an introduction*. A. K. Peters, Ltd., Natick, MA, USA.

DELINGETTE, H. 1998. Towards realistic soft tissue modeling in medical simulation. *Proceedings of the IEEE : Special Issue on Surgery Simulation* (Apr.), 512–523.

ELLENS, M. S., AND COHEN, E. 1995. An approach to c^{-1} and c^{0} feature lines. *Mathematical Methods for Curves and Surfaces*, 121–132.

EUGENE, M., TARDY, J., THOMAS, J. R., AND BROWN, R. 1995. *Aesthetic Facial Surgery*. Mosby, January.

FOREST, C., DELINGETTE, H., AND AYACHE, N. 2002. Cutting simulation of manifold volumetric meshes. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer, Tokyo, Japan, T. Dohi and R. Kikins, Eds., vol. 2489, 235–244.

GANOVELLI, F., AND O'SULLIVAN, C. 2001. Animating cuts with on-the-fly re-meshing. In *Eurographics 2001 - short presentations*, 243–247.

GANOVELLI, F., CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 2000. A multiresolution model for soft objects supporting interactive cuts and lacerations. ??–??

MASUTANI, Y., AND KIMURA, F. 2001. Modally controlled free form deformation for non-rigid registration in image-guided liver surgery. In *Proceedings Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer, Utrecht, NL, W. J. Niessen and M. A. Viergever, Eds., vol. 2208 of *Lecture Notes in Computer Science*, 1275–1278.

MOR, A., AND KANADE, T. 2000. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI).*, Springer-Verlag, vol. 1935, 598–607.

NIENHUYS, H.-W., AND VAN DER STAPPEN, A. F. 2000. Combining finite element deformation with cutting for surgery simulations. In *EuroGraphics Short Presentations*, A. de Sousa and J. Torres, Eds., 43–52.

NIENHUYS, H.-W., AND VAN DER STAPPEN, A. F. 2001. A surgery simulation supporting cuts and finite element deformation. In *Medical Image Computing and Computer-Assisted Intervention*, Springer-Verlag, Utrecht, The Netherlands, W. J. Niessen and M. A. Viergever, Eds., vol. 2208 of *Lecture Notes in Computer Science*, 153–160.

NIENHUYS, H.-W., AND VAN DER STAPPEN, A. F. 2004. A delaunay approach to interactive cutting in triangulated surfaces.

In *Fifth International Workshop on Algorithmic Foundations of Robotics*, Springer-Verlag Berlin Heidelberg, Nice, France, J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds., vol. 7 of *Springer Tracts in Advanced Robotics*, 113–129.

SCHEIN, S., AND ELBER, G. 2005. Discontinous free-form deformation. In *The 12th Pacific Conference on Graphics and Applications (PG)*, 227–236.

SCHNABEL, J. A., RUECKERT, D., QUIST, M., BLACKALL, J. M., CASTELLANO-SMITH, A. D., HARTKENS, T., PENNEY, G. P., HALL, W. A., LIU, H., TRUWIT, C. L., GERRITSEN, F. A., HILL, D. L. G., , AND HAWKES, D. J. 2001. A generic framework for non-rigid registration based on non-uniform multi-level free-form deformations. In *Proceedings in Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, W. J. Niessen and M. A. Viergever, Eds., vol. 2208, 573–581.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. *Computer Graphics 20* (Aug), 151–160.

SELA, G., SCHEIN, S., AND ELBER, G. 2004. Real-time incision simulation using discontinuous free form deformation. In *Medical Simulation: International Symposium, ISMS 2004, Cambridge,MA, USA, June 17-18, 2004*, Springer, S. Cotin and D. N. Metaxas, Eds., vol. 3078 of *Lecture Notes in Computer Science*, 114–123.

SENSABLE TECHNOLOGIES, INC. PHANTOM haptic device. `http://www.sensable.com`

SEYMOUR, N. E., GALLAGHER, A. G., ROMAN, S. A., O'BRIEN, M. K., BANSAL, V. K., ANDERSEN, D. K., AND SATAVA, R. M. 2002. Virtual reality training improves operating room performance: Results of a randomized, double-blinded study. *Annals of Surgery 236*, 4, 458–464.

SHI-MIN, H., HUI, Z., CHIEW-LAN, T., AND JIA-GUANG, S. 2001. Direct manipulation of ffd: Efficient explicit solutions and decomposible multiple point constraints. *The Visual Computer 17*, 6, 370–379.

TAN, H., SRINIVASAN, M., EBERMAN, B., AND CHANG, B., 1994. Human factors for the design of force-reflecting haptic interfaces.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Computer Graphics (Proc. SIGGRAPH'87) 21*, 4, 205–214.

VIGNERON, L. M., VERLY, J. G., AND WARFIELD, S. K. 2004. Modelling surgical cuts, retractions, and resections via extended finite element method. In *Proceedings of the 7th Int. Conf on Medical Image Computing and Computer-Assisted Intervention - MICCAI 2004 (2)*, Springer Verlag, Saint-Malo, France, C. Barillot, D. Haynor, and P. Hellier, Eds., vol. 3217 of *LNCS*, 311–318.

WU, W., AND PHENG-ANN, H. 2004. A hybrid condensed finite element model with gpu acceleration for interactive 3d soft tissue cutting. *Computer Animation and Virtual Worlds (CAVW) Journal 15*, 3-4, 219–227.