A Microsurgery Simulation System

Joel Brown[1], Kevin Montgomery[2], Jean-Claude Latombe[1], and Michael Stephanides[2]

[1] Computer Science Department, Stanford University
[2] Department of Surgery, Stanford University

Abstract. Computer systems for surgical planning and training are poised to greatly impact the traditional versions of these tasks. These systems provide an opportunity to learn surgical techniques with lower costs and lower risks. We have developed a virtual environment for the graphical visualization of complex surgical objects, and real-time interaction with these objects using realistic virtual tools. An application for microsurgical training, in which the the user sutures together virtual blood vessels, has been developed. This application demonstrates many facets of our system, including deformable object simulation, tool interactions, collision detection, and suture simulation. Here we present a broad outline of the system, which can be generalized for any anastomosis or other procedures, and a detailed look at the components of the microsurgery simulation.

1 Introduction

As computer power and graphics capabilities continue to grow, there is an increased focus on surgical simulation as a technique to enhance surgeons' training. In surgical training, which currently requires cadavers or laboratory animals, a simulation option could reduce costs and allay ethical concerns, while possibly decreasing training time. However, for surgical simulation to be useful it must be realistic, including realistic tissue deformation, real tool interactions, and visual accuracy, all performed in real-time.

1.1 Related Work

Research on modeling soft tissue deformation has increased dramatically in the past few years, with focus on physically-based models for simulation. Terzopoulos and Waters [1] argue the advantages of using anatomy and physics rather than just geometry for facial animation, and present a mass-spring model of facial tissue with muscle actuators. Joukhadar and Laugier [2] also use a mass-spring model with explicit integration techniques as the foundation of a general dynamic simulation system, and Baraff and Witkin [3] use masses and springs with implicit integration to simulate cloth. These mass-spring models are characterized by fast computation and simple implementation [13], while providing a degree of physical realism based on linear or visco-elastic tissue models.

More complex finite element models (FEMs) have been used to capture more biomechanical properties of human tissues. [4], [5], and [6] all use FEMs to model facial tissue and predict surgery outcomes. The increased computational demands of finite elements are serious hurdles for real-time simulation. Bro-Nielsen and Cotin [7] and Berkley et al. [8] propose numerical techniques to significantly reduce computation in finite element models. The endoscopic training tool of Kühnapfel et al. [9] describes a system which uses either mass-spring or FEM depending on the situation. The hybrid elastic approach of Delingette et al. [14] combines aspects of both models in a simulation which allows deformation and cutting of tissues in real-time.

There are certainly many other examples of mass-spring and finite element models, as well as several completely different deformable models, too numerous to cite them all here. The consensus is that FEMs can be very biomechanically accurate, but are not always appropriate for large deformations, or for real-time simulation of large geometries. Conversely, determining the proper parameters for mass-spring models can be very difficult, as can be determining proper placement of masses and springs to adequately model an object's volume. Surgical simulation tends to rely more on visual realism than exact deformation (which may be more necessary in the prediction of surgical outcomes, for example), and thus we have focused our research on mass-spring models, with an eye towards computation reduction for displaying complex virtual environments.

The need for suture simulation has been addressed by Delingette [15], and a performance study by O'Toole et al. [16] discusses the validity of using such a simulator to develop surgical skill, although it does not provide any technical details of the actual simulation. More recently, Kühnapfel et al. [10] discuss many different aspects of suture simulation.

Interaction with virtual tools is another necessary component of any realistic simulator, and has attracted a fair amount of recent attention [11, 10, 12]. Surgical tools must not only be modeled accurately, but also pull, push, prod, and cut other objects, often causing deformations by their actions. The first step of simulating such interactions is accurate collision detection, which has been studied extensively [19], although not nearly as much with deformable models. After detection, novel collision handling algorithms will need to be developed to allow for the wide range of surgical actions.

1.2 Description of Microsurgery

Microsurgery is a well-established field which involves the repair of approximately 1mm vessels and nerves under an operating microscope. Using a forceps, the surgeon maneuvers a suture through the ends of two vessels, loops the suture around a second forceps, and pulls it tightly through itself to knot the vessels together. With several such stitches, the severed vessel can be repaired. Microsurgeons typically acquire their initial skills through months of practice in an animal lab, at which point they still require months of supervision in the operating room. Without practice, these skills can quickly degrade.

This paper presents a generalized surgical simulation system and a specialized microsurgery training tool. Real surgical tools, fast and realistic tissue deformation, and a realistic suture simulation are integral components of this tool.

2 System Overview

The *Spring* software system allows any number of objects to be displayed in a 3D virtual world, with complete user control of the view. 3D views are obtained using stereo glasses (CrystalEyes, Stereographics). All software is written in portable C++ and OpenGL, running on Solaris, Windows, and SGI operating systems. Results and screenshots shown here are on a Sun Ultra60 with dual 450 MHz Sparc processors.

Objects are modeled with vertices, edges, and triangles, and can be created programmatically, reconstructed from patient data (e.g. for organ models), derived from geometry data (e.g. for surgical tool models), or imported from a variety of data formats. Surface triangles allow for complex shading, lighting, and texturing, while arbitrary connections of vertices can exist under the surface. The vertices and edges can represent the fixed geometry of rigid objects, the masses, springs, and dampers of viscoelastic deformable objects, or special cases such as the joints and links of articulating objects.

External devices, such as tracking devices and force-feedback devices, provide real time position and orientation input to the simulation (as well as haptic output), for example to control virtual tools. These devices interface with the software to provide interactions such as grabbing, poking, and cutting.

The system also supports parallel processing using multithreading. Two separate threads of execution allow the simulation (deformation and movement of objects) to not conflict with the display (rendering all objects on the screen from the correct viewpoint). In this way, visual updates may occur at 30 Hz, while simulation and interfaces with external devices run as fast as possible.

The setup for microsurgery includes two real surgical forceps instrumented to detect closure and attached to electromagnetic trackers (miniBIRD, Ascension Technology Corporation). The user's translation, rotation, opening, and closing of these forceps directly controls forceps models in the simulation. Using the forceps, models of blood vessels can be grabbed and deformed. A suture (needle and thread) can be manipulated to pierce through and realistically interact with the vessels and the forceps. This is all viewed with stereo glasses to allow the necessary depth perception to complete the task.

3 Soft Tissue Modeling

We represent the volumetric geometry of a deformable object by a 3D mesh (network) M of n nodes N_i ($i = 1,...,n$) connected by links L_{ij}, $i, j \in [1, n], i \neq j$. The nodes and links of M are the vertices and edges of our mesh, grouped into triangles on the surface, but unrestricted below the surface. Each node maps to

Fig. 1. The user controls two real forceps to manipulate the vessels in the simulation

a specific point of the object, so that the displacements of the nodes describe the deformation of the object.

The viscoelastic properties of the object are described by additional data stored in the nodes and links of M. More precisely, a mass m_i and a damping coefficient c_i are associated with each node N_i, and a stiffness k_{ij} is associated with each deformable link L_{ij}. The internal force between N_i and N_j is $\boldsymbol{F}_{ij} = -k_{ij}\Delta_{ij}\boldsymbol{u}_{ij}$, where $\Delta_{ij} = l_{ij} - rl_{ij}$ is the current length of the link minus its resting length, and $\boldsymbol{u}_{ij}$ is the unit vector pointing from N_i toward N_j. In all cases, $\boldsymbol{F}_{ij}$ is a function of the coordinate vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ of N_i and N_j. This representation makes it possible to describe objects that have nonlinear, non-homogeneous, and anisotropic properties.

At any instant of time t, the motion and deformation of M is described by a system of n second-order differential equations, each expressing the motion of an N_i:

$$m_i\boldsymbol{a}_i + c_i\boldsymbol{v}_i + \sum_{j\in\sigma(i)} \boldsymbol{F}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) = m_i\boldsymbol{g} \tag{1}$$

where $\boldsymbol{x}_i$ is the coordinate vector of N_i, $\boldsymbol{v}_i$ and $\boldsymbol{a}_i$ are its velocity and acceleration vectors, respectively, and $m_i\boldsymbol{g}$ is the gravitational force. $\sigma(i)$ denotes the set of the indices of the nodes adjacent to N_i in M. Some nodes may be fixed in space, or directly manipulated by surgical tools, in which case their positions are computed directly rather than via 1.

To solve the dynamics of the system, we have implemented several numerical integration techniques to solve (1), including Forward Euler, and second and fourth order Runge-Kutta solvers. These are all based on advancing the simulation by an amount of time Δt, and using the position, velocity, and force information for each node at time t to find positions and velocities at time $t+\Delta t$.

A simplified "quasi-static" algorithm is based upon the assumption that the velocity of user-displaced nodes is small enough that the mesh achieves static equilibrium at each instant. This assumption is reasonable for objects whose deformations are highly damped, as is the case for many human-body tissues. Thus we can neglect dynamic inertial and damping forces, and compute the current shape of M by solving this system of equations:

$$\sum_{j \in \sigma(i)} \boldsymbol{F}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) - m_i \boldsymbol{g} = \boldsymbol{0} \tag{2}$$

The reduced computation can be a significant savings in the many situations where the quasistatic assumptions are appropriate.

We let I be the set of indices of all variable nodes (those which are not fixed in space or directly grasped by the user), and let $\boldsymbol{f}_i$ be the total force acting on each variable node, $\boldsymbol{f}_i = \sum_{j \in \sigma(i)} \boldsymbol{F}_{ij} - m_i \boldsymbol{g}$. An iterative algorithm for solving (2) at real-time animation rates is as follows:

1. Read and set the position of every user-displaced node
2. Repeat until 1/30 sec. has elapsed
   (a) For every $i \in I$, update $\boldsymbol{f}_i$ based on (2), $\boldsymbol{f}_i \leftarrow \sum_{j \in \sigma(i)} \boldsymbol{F}_{ij} - m_i \boldsymbol{g}$
   (b) For every $i \in I$, $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \alpha \boldsymbol{f}_i$

Ideally, the value of α is chosen as large as possible such that the iteration converges, and the values $\boldsymbol{f}_i$ approach $\boldsymbol{0}$ before each redraw. This choice typically requires some trial-and-error experiments. By rendering in a separate thread, the entire time interval is used for computing the equilibrium positions $\boldsymbol{x}_i$ of the nodes.

The blood vessels in our simulation are modeled as double-hulled cylinders, with the inner and outer cylinders representing the thickness of the vessel. Each cylinder consists of several layers of nodes, with the layers evenly spaced, and each layer consisting of several nodes evenly spaced around a circle. Each node is connected by edges (springs) to its neighbors within a layer and in neighboring layers. There are also connections between the inner and outer cylinders, which provide torsional stability, preventing the vessel from twisting around its long axis. The end layers of each vessel are fixed in space, representing the fact that the vessels are clamped down during surgery, and only a portion of their length can be manipulated. Figure $2a$ shows one smooth-shaded and one wireframe vessel.

As the user displaces individual nodes of the vessels, the quasi-static algorithm described above is used to calculate the deformation. Figures $2b$ and $2c$ shows some examples of vessel deformation.

4 Simulation of the Suture

The suture is deformable, but is not viscoelastic, so the above deformation techniques based on mass-spring models are not applicable. Instead it should behave as a needle and thread, which can stretch minimally if at all, and has a free-form shape which is affected by gravity and direct contacts. To achieve realistic deformation, the suture is modeled as an articulating object: many short, linear

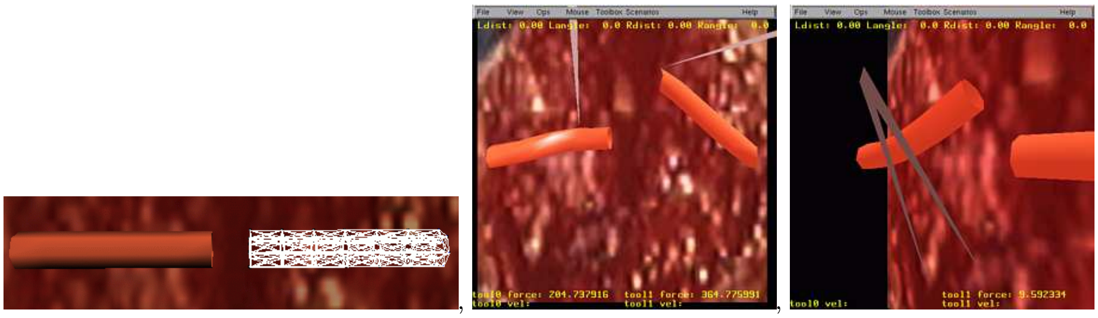

Fig. 2. (a) Smooth-shaded and wireframe vessels, (b),(c): Vessels are deformed with forceps

segments are connected together, and the only constraint between two neighboring segments is that they share an endpoint. Put another way, the suture is a sequence of nodes (joints) and edges (links), with each edge a fixed length, and an arbitrary angle allowed at each node between neighboring edges. By keeping the angle between the first few edges fixed, we can model a rigid needle at one end of the suture.

To model the motion of the suture, a constraint-based system is used. Any node of the suture may be constrained by another object in the system. For example, one node might be grasped by a forceps, and thus its position is constrained by the forceps. If the suture has pierced through a vessel, a node will be constrained by the position of the vessel. Finally, if the suture is draped over another object, nodes will be constrained by that object.

The motion is then calculated in a "follow-the-leader" manner as follows: a constrained node N_i is moved by the constraining object from $\boldsymbol{x}_{i.old}$ to $\boldsymbol{x}_{i.new}$. Its neighbor N_{i+1} then computes its new position $\boldsymbol{x}_{i+1.new}$ as the point a distance d along the line from $\boldsymbol{x}_{i.new}$ to $\boldsymbol{x}_{i+1.old}$, where d is the (fixed) length of the edge connecting N_i and N_{i+1}. The same is done for node N_{i-1}. This motion is propagated up and down the suture to N_{i+1}, N_{i+2}, ... and N_{i-1}, N_{i-2}, ... until the next constrained node or one end of the suture is reached. For nodes between two constrained nodes N_i and N_j, the preceding algorithm will compute two preliminary results, propagating from N_i to N_j, and from N_j to N_i. These results are averaged to give the final position.

Certain constraints are designated as *soft*, such as where the suture is piercing a vessel or draped over another object, whereas the forceps grabbing the suture is a *hard* constraint. The distinction is that the suture can slide over/through a soft constraint, changing which node of the suture is constrained. It may be the case that two constraints move in opposing directions and cause the suture to stretch between them. In that case, the suture will slide through a soft constraint to decrease the stretch, but will break if between two hard constraints (in real surgery, it is not difficult to break the suture by pulling with both forceps in opposite directions). Additionally, if the suture is pierced through a vessel and is pulled on both ends, the suture will pull the vessel, causing it to deform as in Fig. 3*a*. Figure 3*b* shows the suture pulling together the two vessels.

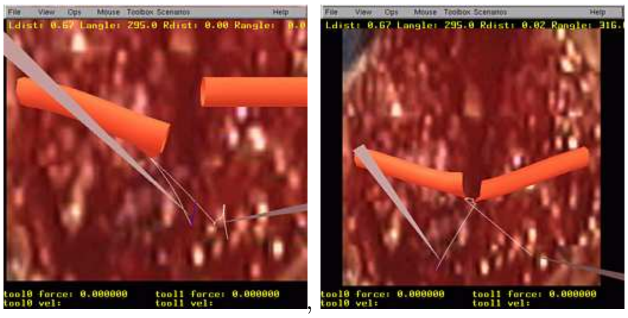

Fig. 3. (a) Suture pulling down on a vessel, (b) Suture pulling two vessels together

5 Collisions and Interactions

Almost all object interactions depend at some level on collision detection [19]. Grabbing is achieved by finding nodes colliding with the tip of the grabbing object (e.g. forceps). Piercing the vessel requires finding a collision between the needle edge and a vessel face. Draping the suture around another object also involves edge to face collisions (see Fig. 4), and draping it around itself requires edge to edge self-collisions. Other interactions modeled by the system (although not specifically in the microsurgery simulation) include prodding one object with another (face to face collisions), and cutting one object with another (edge to face collisions).

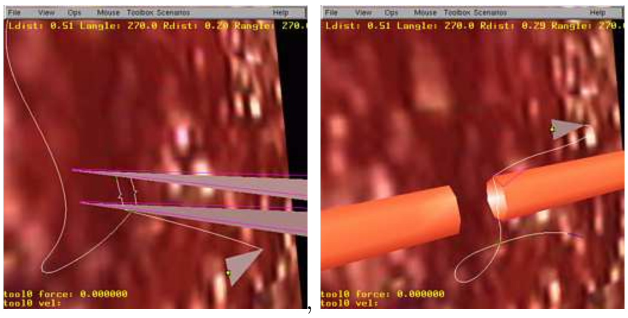

Fig. 4. (a) Suture wrapped around forceps, (b) Suture colliding with self and vessel end

The scheme we use for collision detection and distance computation is based on Quinlan's bounding sphere hierarchy [17]. This algorithm was extended by Sorkin to efficiently allow for collisions between deforming objects [18]. We can comfortably find all collisions between forceps, vessels, and suture (including internal edge to edge collisions in the suture) at every redraw, without affecting the animation rate of the simulation.

6 Conclusions and Future Work

By combining tool interactions, tissue deformation, collision detection, and high-resolution graphics, we have created a microsurgery simulation which has already been tested by over 100 plastic and reconstructive surgeons, and deemed realistic and potentially very useful. To make this training tool a reality, experimental verification is needed. Currently the angle and position of vessel piercing is measured. With this and other measurements, we can give feedback as to the quality of the virtual repair. We must then test how practicing with the simulation relates to future quality of real vessel reapirs.

References

1. D. Terzopoulos and K. Waters. Physically-based Facial Modelling, Analysis, and Animation. *The Journal of Visualization and Computer Animation* Vol 1: 73-80, 1990.
2. A. Joukhadar and C. Laugier. Dynamic Simulation: Model, Basic Algorithms, and Optimization. In *Robotic Motion and Manipulation, WAFR 1996*.
3. D. Baraff and A. Witkin. Large Steps in Cloth Simulation. In *Proc. of ACM Siggraph'98*, pages 43-52, 1998.
4. R. Koch, M. Gross, F. Carls, D. von Büren, G. Fankhauser, and Y. Parish, Simulating facial surgery using finite element models. In *SIGGRAPH 96 Conference Proceedings*, 421-428, 1996.
5. E. Keeve, S. Girod, P. Pfeifle, B. Girod. Anatomy-Based Facial Tissue Modeling Using the Finite Element Method. In *Proceedings of IEEE Visualization '96*, San Francisco, CA, Oct. 27 - Nov. 1, 1996.
6. S. Pieper, D. Laub, and J. Rosen. A Finite-Element Facial Model for Simulating Plastic Surgery. In *Plastic and Reconstructive Surgery*, Vol. 96, No. 5, 1100-1105, October 1995.
7. M. Bro-Nielsen and S. Cotin. Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation. *Computer Graphics Forum*, 15(3):57-66 (Eurographics '96), 1996.
8. J. Berkley, S. Weghorst, H. Gladstone, G. Raugi, D. Berg, and M. Ganter. Fast Finite Element Modeling for Surgical Simulation. *Proceedings of Medicine Meets Virtual Reality 1999*, 55-61, 1999.
9. U. Kühnapfel, H. K. Çakmak, H. Maaß. Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers & Graphics*, 24(2000) 671-682.
10. U. Kühnapfel, H. K. Çakmak, H. Maaß, S. Waldhausen. Models for simulating instrument-tissue interactions. *Medicine Meets Virtual Reality (MMVR 2001)*, Newport Beach, CA, January 27, 2001, *http://www-kismet.iai.fzk.de/KISMET/pdf/MMVR_01_UK_Tutor.pdf*

11. K. Montgomery. On Implementing Fundamental Manipulations in Surgical Simulators. *Medicine Meets Virtual Reality (MMVR2001)*, Newport Beach, CA, January 25, 2001, *http://biocomp.stanford.edu/presentation/MMVR01-Interactions.ppt.htm*
12. C. Basdogan. Simulation of Instrument-Tissue Interactions and System Integration. *Medicine Meets Virtual Reality (MMVR2001)*, Newport Beach, CA, January 27, 2001, *http://eis.jpl.nasa.gov/~basdogan/Tutorials/MMVRTuto01.pdf*
13. S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62-73, January-March 1999.
14. S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8):437-452, 2000.
15. H. Delingette. Towards realistic soft tissue modeling in medical simulation. *Proceedings of the IEEE : Special Issue on Surgery Simulation*, pages 512-523, April 1998.
16. R. O'Toole, R. Playter, T. Krummel, W. Blank, N. Cornelius, W. Roberts, W. Bell, and M. Raibert. Measuring and developing suturing technique with a virtual reality surgical simulator. *Journal of the American College of Surgeons*, 189:1 : 114-127, July 1999.
17. S. Quinlan. Efficient Distance Computation Between Non-Convex Objects. *Proc. IEEE Int. Conf. On Robotics and Automation*, 3324-3329, 1994.
18. S. Sorkin. *Distance Computing Between Deformable Objects*. Honors Thesis, Computer Sc. Dept., Stanford University, June 2000.
19. M. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *Proceedings of IMA Conference on Mathematics of Surfaces*, 1998.