Toward Modeling of a Suturing Task

Matt LeDuc, Shahram Payandeh and John Dill
Experimental Robotics and Graphics Laboratory
School of Engineering Science
Simon Fraser University
Burnaby, BC V5A 1S6, Canada
{mleduc/shahram/dill}@cs.sfu.ca

Abstract

In this paper we present our initial work on simulating suture and suturing using mass-spring models. Various models for simulating suture were studied, and a simple linear mass-spring model of the suture was determined to give good performance. A novel model for pulling a suture through a deformable surface is presented. By connecting two separate surfaces through the suture, our model can simulate a suturing task. The results are shown using software we developed that runs on a standard PC and models the action of a suturing device used in minimally invasive Laparoscopic surgery.

1. Introduction

In this paper we attempt to model a suture, and create a simulation of a suturing task realistic enough to use in a surgical training environment, and fast enough to run on a desktop computer. One of our main goals is for the system to run on PC hardware, e.g. a Pentium III system with an Nvidia geForce 2 video card. Such a goal is difficult to achieve, since simulating deformable objects and performing collision detection are both computationally intensive. However, for development of a surgical training environment, e.g. our Laparoscopic Training Environment (LTE), virtual representations need not be extremely precise. They only have to be accurate enough for a trainee to gain the required dexterity and hand-eye coordination. In this paper we model a suture and a simple deformable object. We develop and describe a preliminary task in which we use the suture, complete with a simulated Endo Stitch suturing device that the operator can use to stitch together two deformable objects.

Many groups have been working on surgical simulation in general ([1], [2], [3], and [4]), and the specific task of simulating suturing ([5], [6], and [7]). Measuring surgeons' performance using a simulation was investigated in [5]. However, the paper focused on the initial penetration of the object with the suturing needle, and did not consider the entire suturing process. A group at Rice University took the opposite approach in [6]. They focused only the realistic simulation of a suture and its behavior, while not looking at the actual suturing task. Their paper describes a method for simulating a suture using a spline of linear springs and large overlapping nodes. Although their method gives up some speed and stability, they are able to tie various knots in the suture material.

In [7] a system was designed for training surgeons in the task of suturing blood vessels. Blood vessels were simulated using mass-spring systems while the suture itself was simulated using rigid links of a fixed length. In [8], this same group designed a software framework that supports many different kinds of surgical tasks. Unrelated to surgery simulation, but using similar technology, is research like that done in [9], where various legless animals were simulated using mass-spring systems. This method could possibly be used to create realistically moving organs, such as the heart and lungs, in surgical simulations.

This paper presents an initial novel approach for simulating suture and the suturing task, where the suture and needle is being passed between two tissues in order to connect them. The paper is organized as follows. Sections 2 and 3 describe the deformable models we used to represent the objects in our simulation (suture, and tissue), while section 4 describes the algorithm used to simulate the suturing. Section 5 outlines a demonstration we developed using the techniques described, while Section 6 discusses possible directions for our future research.

2. Deformable Objects

Triangular surface meshes represent the rigid objects in our virtual environment, such as the "glass box" and Endo Stitch device.

Our deformable models are mass-spring models. Mass-spring models, along with finite-element models, are well known ways of simulating deformable objects, [6], [7], and [8], so we limit our discussion to those aspects especially relevant to our development here.

Each node is a mass point and each edge is a spring joining two mass points. Each edge spring when stretched or compressed applies a force of

$$F = K_e * Dir(currentLength - restLength) / restLength$$

on the nodes, where K_e is the spring constant and Dir is the unit vector pointing from the node whose force is being calculated, to the other node of the spring. K_e is the same for all edge springs in the model.

2.1. Home Springs

Using only a mass-spring surface model, one could not construct 3D deformable objects that could be compressed and stretched, since they would not return to their initial shapes after deformation. One approach to solving this problem is to create an internal structure of springs to give the surface the support needed to maintain, and return to its initial shape after being deformed. For example, this method is used to model blood vessels in [7]. Although it proves effective and stable for small models with small displacements, with more complicated objects or large deformations, the object can easily become unstable or permanently tangled.

To address this problem, our models use "*home springs*" connected to each node. These home springs connect each node to a fixed location in 3D space (the original location of the node), and maintain the connected vertex in its undisplaced position through the creation of an internal force proportional, but of opposite direction, to the displacement of the node. As a result, when the object has been deformed, for example by an interaction with another object, after the interacting object has been removed, it will be pulled back to its original shape (figure 1, a-c).

We have used this method before in an early phase of our LTE as well as in a surface mesh subdivision model in [10]. It is an efficient solution since the force applied by each home spring to its connected node is simply calculated as $F = K_h(\text{hom}ePosition - currentPosition)$. This equation consists of only a vector subtraction, and scalar multiplication, and is therefore much faster than the one used for the edge-springs, which involves a square-root operation. Since the number of home springs in a surface model will be proportional to the number of edge springs, this model adds only a small constant amount of computing over the mass-spring surface model.

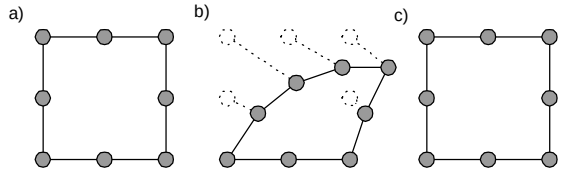

Figure 1. Deformation and restoration of a model containing home springs

2.2. Node-position Integration Method

To solve for the deformed state of the object we use Euler's method to integrate the positions of the nodes under a quasi-static approximation to Newtonian physics (similar to the method mentioned in [7]). In this method, the velocity of a node at a given point in time is calculated only from the forces acting on the node at that instant, and does not include the velocity at the previous time step (*i*-1).

In a standard mass-spring model the position of each node is integrated according the to the following equations, where M is the node's mass, B is a damping constant, *dt* is the timestep, and F_i, V_i, and P_i represent the force acting on the node, the node's velocity, and the node's position, all at time step *i*.

$$F_i = \sum(\text{spring forces}) - BV_{i-1} \qquad eq.\ 1$$

$$V_i = V_{i-1} + \frac{F_i}{M}dt \qquad eq.\ 2$$

$$P_i = P_{i-1} + V_i dt$$

Combining equations 1 and 2, we get

$$V_i = V_{i-1} + \frac{\sum(\text{spring forces}) \ - \ BV_{i-1}}{M}dt$$

If we assume the node mass is relatively small this simplifies to:

$$V_{i-1} = \frac{\sum(\text{spring forces})}{B} \qquad \text{eq. 3}$$

The advantages in using this quasi-static method are speed and simplicity. Since there are fewer calculations, it runs faster (8-10% in our application), and also allows the mass attribute *M* to be left out of calculation.

3. Suture Model

The suture uses the same deformable model data structure that we use for the surface of the objects. The difference is that instead of creating a 2D mesh in 3D space, the nodes are simply arranged linearly, one after another, and joined together with edge-springs (see figure 2). The result is a 1D suture in 3D space.

Because the suture must be able to move within the scene, its home-spring constant K_h is set to zero. We also want the suture to behave realistically under the influence of gravity, so a constant gravitational force is added to each node).

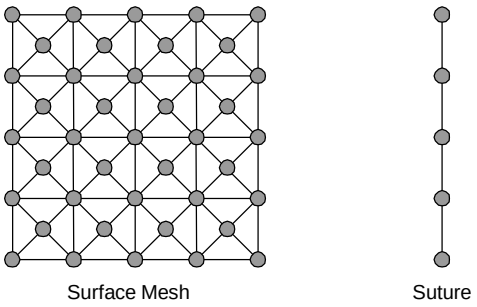

Figure 2. Surface mesh and Suture models.

We investigated several other possible representations of the suture, involving various forms of springs and dampers [11], see figure 3. The first, and simplest one, was simply masses connected together by springs and involved no damping. The second model added dampers running between the masses. Three more complicated, and more realistically behaving, models involving torsion spring, torsion dampers, and viscous damping effects were also implemented. We chose to use the first model for the suture in this simulation. This model was very fast to calculate, but originally behaved unrealistically due to the lack of damping. Using our quasi-static method for integrating the position of the model's nodes, a global viscous damping effect (eq. 3) is introduced without adding to the complexity of the calculations and slowing the simulation down.

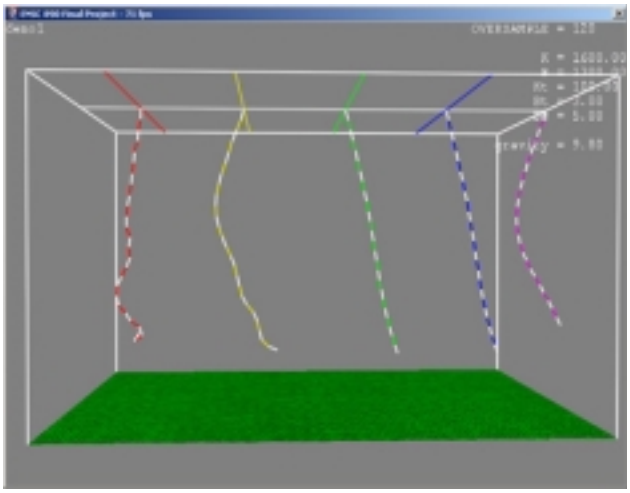

Figure 3. Various suture models, each with different construction and behavior.

3.1. Suture Rendering

Since the nodes of the suture lie in a linear chain, a commonly used method is to render the suture as line segments. This is fast and simple, but it would not be the same rendering method used by the triangle-based objects, and would therefore look very different.

To avoid this problem, we chose to render the suture by creating a cylinder that contains the same number of sections as there are segments in the suture. We then reposition this cylinder over of the suture before each frame is rendered. This newly defined shell is then rendered instead of the suture itself. An illustration of the process can be seen in figure 4 below. Since the suture is now rendered using a triangle model, it can now undergo the same lighting calculations, and have a similar appearance.

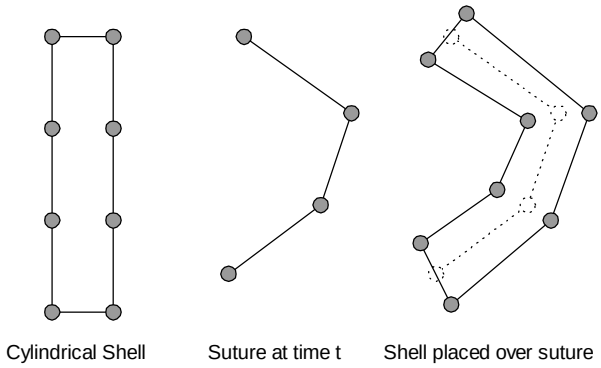

Figure 4. Suture rendering

4. Simulated Suturing

To make the problem of simulating suturing more manageable, we have chosen to ignore for now the problem of inserting a needle into a deformable object, and only deal with a suture that has already passed through the object.

4.1. Basic Suturing algorithm

In real suturing, the needle passes through an object, creating a hole through which the thread is pulled. As long as the forces on the suture are small, friction between the suture and object will tend to prevent the suture from sliding through the hole, so the suture will pull the object along with it. Simulating suturing by both creating a small hole in the triangularly modeled deformable object, and simulating the friction forces between it and the suture would be overly complex for the purpose of a training environment.

Instead, we model the above effects by treating one of the nodes of the object as a hole, and connecting this node to one of the nodes of the suture. This can be seen in figure 5a, where the filled circles are the nodes of the object, and the hollow circles are nodes of the suture. In figure 5a, there is no force being applied to the suture. In

figure 5b, a force is applied. This force pulls the suture toward the upper right. Since the node of the suture is joined to a node of the object, the two move together as one, and the rest of the object gets pulled along with it.

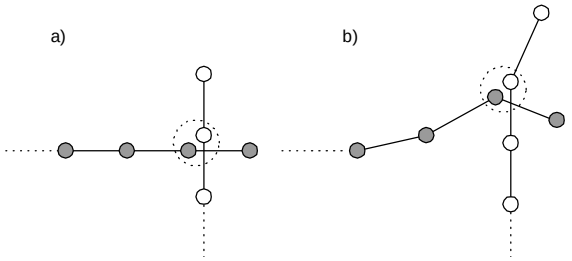

Figure 5. Simulation of the suture running through a small hole in the object.

When the object gets pulled along due to the friction between it and the suture, there is a limit to how far it will move. Eventually the forces on the object, which are created by the solution of the mass-spring equations, will become large enough that the friction force cannot prevent the object from sliding down the suture.

In figure 6a, node N of the object model (the hole through which the suture has been pulled) is being pulled down by its neighboring nodes; however, the force being applied to it from the thread balance these downward forces. Once the suture has been pulled too far, and the object stretched too long, the required force from the suture to the object in order to keep it from sliding, will be greater than the friction between them. To simulate this sliding, node N is detached from node S_0 of the suture, and reattached to node S_1. If the suture continues to be pulled, then node N will continue to slide down the thread (figure 6b), creating the impression the suture is slipping through a hole.

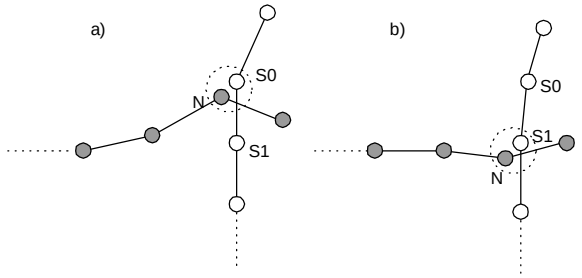

Figure 6. Slipping of the deformable object down the suture.

4.2. Multiple Slipping

During suturing, two or more objects will be pulled together by a suture. In figure 7a a suture is shown between two pieces of modeled tissue. In order to stitch the two pieces together the suture will first pass through the left object, and then the right. Using the suturing algorithm of 4.1. can lead to the situation shown in figure 7b where two object nodes will both be attached to the same suture node. In the present algorithm there is no inter-object, or self-collision detection between the deformable models. Hence, a method is needed to ensure that the two objects on the suture are not able to slide past each other. For example, in figure 7c the right side object will be under more strain than the one on the left, and it will want to slide; however, because it lies above the left object on the suture, it can not slide without pulling the left piece with it.

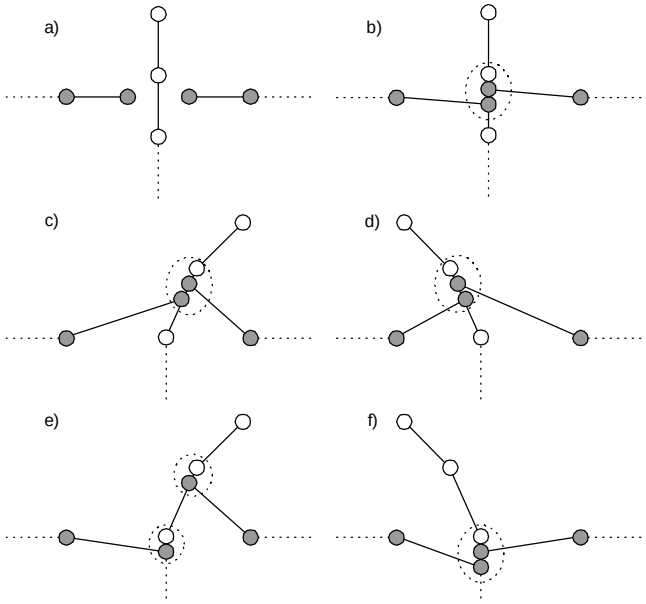

Figure 7. Multiple objects sliding down the suture.

To handle this situation, for each suture vertex we store an ordered list of object nodes that are attached. This linked-list approach allows us to maintain the order in which the object nodes were pierced by the suture. This information allows us to easily handle situations such as those shown in figures 7c and 7d. In figure 7c the left object is under more strain that the one on the right, and it will slip down the suture leaving the other object node behind (figure 7e). In figure 7d the right object is under more strain; however, it cannot slip without pulling the right object with it (figure 7f). This can happen only when the force on the right attached object node is large enough to overcome the friction between itself and the suture, and the combined force of the attached object nodes is enough to overcome the combined friction between the nodes and the suture. If this is not the case, then the object nodes will not slide.

5. Endo Stitch Suturing Task

For our simulation of a suturing task, we modeled an Endo Stitch device and used that to perform the suturing. The end of this device has two jaws and can, through the activation of a mechanical switch, pass a needle between them. With the needle on one of the jaws, the surgeon can pierce the tissue. By closing the jaws and activating the switch, the needle will be passed to the second jaw, pulling the suture through the puncture. This procedure continues until the stitch is formed. We simplified the operation of the virtual Endo Stitch device slightly: a single key press on the computer's keyboard will close the jaws, pass the needle across, and then open the jaws again. In the present preliminary simulation of the suturing task, the device is either activated, resulting in the needle passing through the object, or it is not. Thus we have avoided the need to model the interaction between the needle and object.

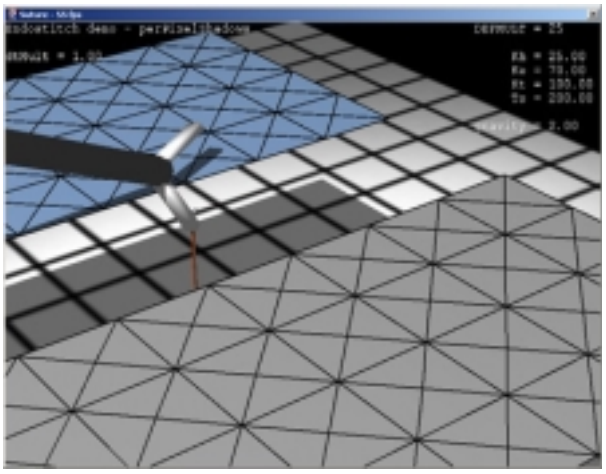

Figure 8. Jaw with needle is under the tissue.

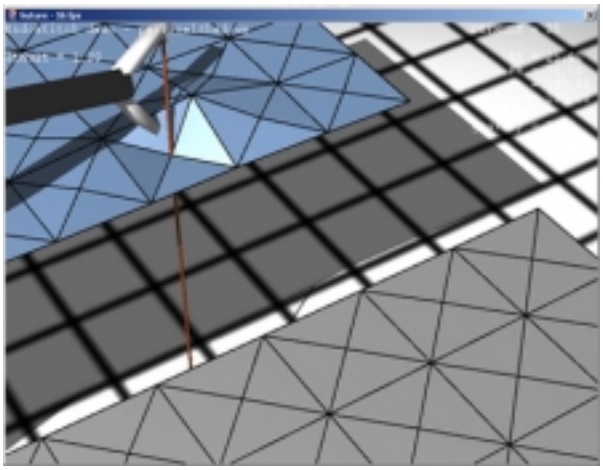

Figure 9. Needle has passed from one jaw to the other, pulling suture through first object.

In the simulation we define two simple objects as flat meshes, one blue, one grey. To suture the two together, one positions the device so that the two jaws of the device are on opposite sides of an object (figure 8).

Pressing the keyboard key will pass the needle through the deformable object to the other jaw. Raising the device pulls the suture through the object (figure 9 Performing this same process on the other object, and pulling on the suture slightly, will bring the two objects together (figure 10). To continue stitching the objects together, the process is repeated (figure 11).

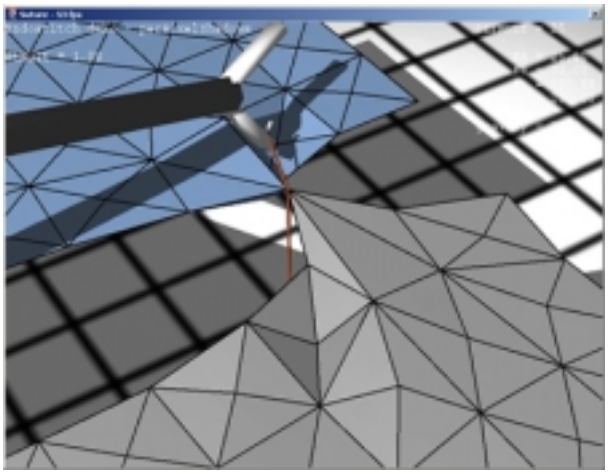

Figure 10. Objects pulled together after passing suture through second object,.

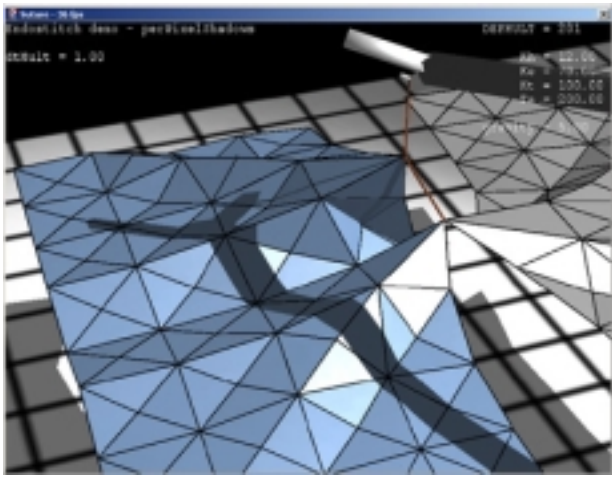

Figure 11. Continuation of suturing procedure.

6. Future Work

Many different directions could be explored in future work on this simulator. Eventually we would like to be able to simulate suturing using a needle, a suture, and a pair of grippers. Performing suturing in this way will create several problems, such as how to grasp, move around, and release the needle using the gripper, and how to simulate the needle interacting with, and eventually penetrating, the deformable object.

Another area of improvement is in collision detection. In the future it may be possible to have the tool be able to touch and deform the objects being sutured, prevent the suture from hanging down through the objects, and ideally have the suture and objects perform self-collision detection. Having the suture collide with itself could lead to the ability to tie knots in the thread, and/or perform more complicated types of suturing.

Last but not least is force feedback. In the near future we intend to integrate a haptic force feedback device into our design of a system for training laparoscopic surgeons. This will not only provide the user with force feedback, but since the device has a handle identical to those used in laparoscopic surgery, it will also provide a more realistic interface to the simulation.

References

[1] P. Gorman, J. Lieser, W. Murray, R. Haluck, and T. Krummel, "Evaluation of Skill Acquisition Using a Force-Feedback, Virtual Reality-based Surgical Trainer", *Proc. Medicine Meets Virtual Reality 1999*, IOS Press, 1999, pp 121-123.

[2] J. Berkley, S. Weghorst, H. Gladstone, G. Raugi, D. Berg, and M. Ganter, "Fast Finite Element Modeling for Surgical SImulation", *Proc. Medicine Meets Virtual Reality 1999, ISO Press*, 1999, pp. 55-61

[3] H. Delingette, "Towards realistic soft tissue modeling in medical simulation", *proc. of the IEEE: Special Issue on Surgical SImulation*, April 1998, pp. 512-523

[4] U. Kuhnapfel, H. Cakmak, H. MaaB, "Endoscopic surgery training using virtual reality and deformable tissue simulation", *Computers & Graphics 24 (2000)*, pp. 671-682

[5] Robert V O'toole, Robert R Playter, Thomas M Krummer, William C Blank, Nancy H Conelius, Webb R Roberts, Whitney J Bell, Marc Raibert "Measuring and Developing Suturing technique with a Virtual Reality Surgical Simulator", *Journal of the American College of Surgeons*, July 1999, pp. 114-27

[6] Andrew Ladd, "Simulated Knot tying", Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington DC

[7] Joel Brown, Kevin Montgomery, Jean-Claude Latombe, and Michael Stephanides, "A Microsurgery Simulation System", *Medical Image Computing and Computer Aided Interventions*, The Neatherlands October 2001

[8] K. Montgomery, C, Bruyns, J. Brown, S. Sorkin, F Mazzela, G. Thonier, A. Tellier, B. Lerman, A. Menon, "Spring: A General Framework for Collaborative, Real-time Surgical Simulation", *Medicine Meets Virtual Reality*, IOS Press, Amsterdam, 2002

[9] Gavin S.P. Miller, "The Motion Dynamics of Snakes and Worms", *Computer Graphics*, Volume 22, November 4 1998, pp169-178

[10] Jian Zhang, Shahram Payandeh and John Dill, "Haptic Subdivision: an Approach to Defining Level-of-detail in Haptic Rendering", *10th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, IEEE Computer Society, pp. 201-208

[11] Matt LeDuc "Suture Simulation", Internal Report, Simon Fraser university, School of Engineering Science, May 2002